AML HW 2: Written exercises

Wednesday, September 30, 2020 10:52 AM

① Max Likelihood & KL Divergence

② 10,000,000 widgets made/year

95% accurate → $P(pos|def) = P(neg|!def) = 0.95$

$P(def) = \frac{1}{100,000}$

(a) $P(def|pos) = \frac{P(pos|def)\,P(def)}{P(pos)} = \frac{P(pos|def)\,P(def)}{P(pos|def)P(def) + P(pos|!def)P(!def)}$

$= \frac{0.95(0.000001)}{0.95(.000001) + 0.05(1-.000001)}$

$= 1.9 \times 10^{-4} = 0.00019$

$= 0.019\%$

(b) Throw away all pos tested; Find $P(!def|pos)$ · # thrown away /year

$P(!def|pos) = \frac{P(pos|!def)P(!def)}{P(pos)}$

$= \frac{0.05(1-0.000001)}{0.05(1-0.00001) + 0.95(0.000001)} = 0.0500009$

$= 0.999981$

thrown away/year = $10 \times 10^6 \cdot P(pos)$

$= 10 \times 10^6 (0.0500009)$

$= 500,009$

$0.999981(500,009) = 499,999$

defective shipped/year = # shipped · $P(def|neg)$

$P(def|neg) = \frac{P(neg|def)P(def)}{P(neg)}$ → $P(neg|def)P(def) + P(neg|!def)P(!def)$

$= \frac{0.05(0.000001)}{0.05(0.000001) + 0.95(1-0.000001)}$

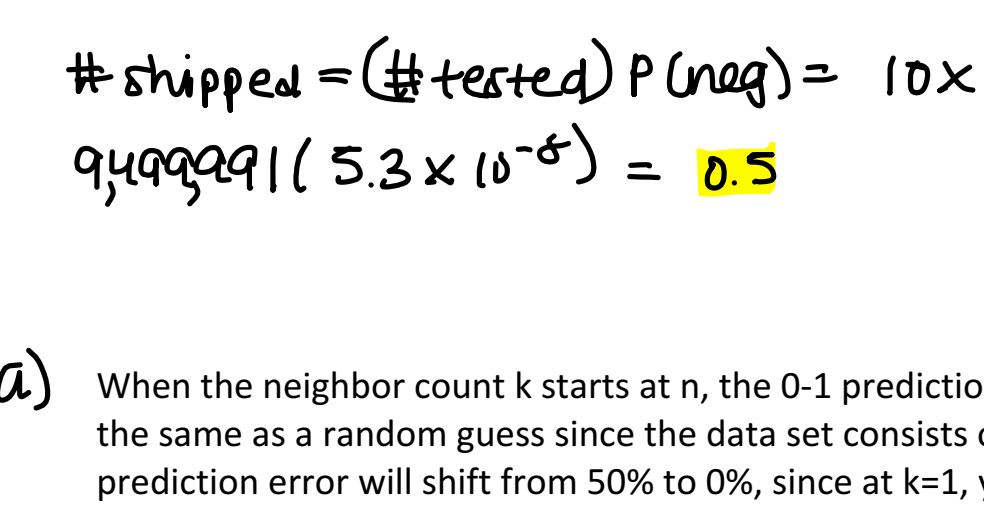
$= 5.263 \times 10^{-8}$

shipped = (# tested) $P(neg) = 10 \times 10^6 (0.9499991) = 9,499,991$

$9,499,991(5.3 \times 10^{-8}) = 0.5$

③ (a) When the neighbor count k starts at n, the 0-1 prediction error will initially be 50%, as this is essentially the same as a random guess since the data set consists of half 0s and half 1s. As k decreases to 1, the prediction error will shift from 50% to 0%, since at k=1, you will be comparing the data point (xi,yi) to itself.

(b) The average 0-1 prediction error in this case essentially shows the case of overfitting to underfitting as k increases from 1 to n. When k is very small, the prediction error will be quite high due to overfitting on our training data. Then, as k increases slightly, the prediction error will decrease, as we are neither underfitting nor overfitting to our training data, yielding an optimal k value for our classification model. As k increases too much, however, the model will be underfit and the prediction error will increase further so that we again would not be able to predict our test data well.



(c) When considering how many different numbers of folds N to use for kNN, we know that for every iteration, you have to train your data on N buckets, each with 1/N the amount of samples, N times, repeating this process multiple times to find a k value which produces an optimal fit. This means that you have to iterate through the full dataset at least N times, but more likely some multiple of N times to find an appropriate k value. With large datasets, the computational requirements significantly increase in terms of time to run for increasing N values. In terms of validation accuracy, we want to pick an N such that we are neither overfitting nor underfitting our data, so we don't want too low or too high of a value of N. Ultimately, due to computation time as seen in this lab, lower numbers of N still can yield high average validation accuracies, so an N of 3 or 5 should suffice. A modification to this algorithm could incorporate a weighted k-nearest neighbors algorithm where, rather than treating all k neighbors as equal samples to fit a model to, you could make only a small subset (say k = 5) have significantly more weight than say the last few k neighbors. Alternatively, you could make the weights on a logarithmic scale such that each k neighbor is weighted differently with closest neighbors having the most weight and farther neighbors having the least weight. This way, all neighbors are still contributing to the model fit but in a more helpful way.