Explanation for KNN, Multinomial Naive Bayes and Guassian Niave Bayes through the lens of a Spam Filter

Cressida L

April 2022

Contents

1 Introduction

The purpose of this project was to gain an understanding of machine learning, and create a tangible project that was somewhat security related! I decided to go with a spam filter, as something like this was something I had never thought too much about, and sending dodgy emails is a way attackers often employ get into systems. This document explains the mechanics of the methods very simply, and does not go into the optimisations and what data sets would benefit from each type of method.

2 Challenges

Type 1 and type 2 errors happened quite a lot, and they were the main type of error I encountered. There isn't actually a way to combat these with the code; the only way to lower the percentage of these errors was to give the algorithms more data points, which would increase the amount of time needed to run it.

3 Explanation of Methods

3.1 KNN

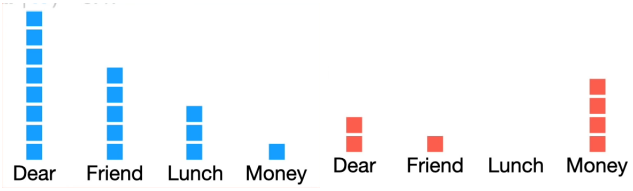
Accuracy: 74%
Knn is shorthand for Kth nearest neighbour. This method looks at the kth most similar data points to the one you want to classify, and will give you a classification depending on the classification of the nearest neighbours. E.g. we have a fruit of some sort, and we want to classify it. We look at maybe the 11 most similar data points from an already classified dataset and see that 5 of the neighbours are apples, but 6 of the neighbours are oranges. Because the majority of the neighbours are oranges, we would classify the fruit as an orange. It is important to note that as we increase k, the nearest data points begin to reflect the dataset as a whole, so it we try to not make k too large. This means the quality and size of our dataset is very important for this method. As k decreases, we have fewer and fewer data points to draw a classification from, so we also don't want a k that is too small. We also want k to be an odd number in this case, so there is never an even split between our two classifications (oranges and apples).

To measure similarity in the case of emails, we look at the frequency of words in the given email and compare it to the frequency of those words in our dataset. We sort the list of training emails, and pick out the kth most similar emails to return a classification.

This method was the most time consuming to run. It would be more accurate with more data, but it would also take even longer to run.

3.2 Multinomial Naive Bayes

Accuracy: 94%
In this method, you look at how frequently words appear in ham and spam emails your dataset and then compare given email to your dataset's word frequencies. You start off by creating a histogram of the words in the dataset, which could look like this:



Let's say the blue histogram is for ham, and the red one is for spam. The probabilities for each of the words are:

$$
\begin{aligned}
p(Dear \mid Ham) &= 0.47 \\
p(Friend \mid Ham) &= 0.29 \\
p(Lunch \mid Ham) &= 0.18 \\
p(Money \mid Ham) &= 0.06 \\
p(Dear \mid Spam) &= 0.29 \\
p(Friend \mid Spam) &= 0.14 \\
p(Lunch \mid Spam) &= 0.00 \\
p(Money \mid Spam) &= 0.57
\end{aligned}
$$

We figure this out by counting the boxes up for each word and dividing by the total number of boxes of that colour. Let's analyse the email "Dear Friend", which we can clearly see is probably spam. To calculate the probability of it being ham or spam, we use these formulae:

$$p(Ham|w_1, w_2, w_3 \ldots, w_n) = p(w_1 \mid Ham) \times p(w_2 \mid Ham) \times p(w_3 \mid Ham) \times \ldots \times p(w_n \mid Ham)$$

$$p(Spam|w_1, w_2, w_3 \ldots, w_n) = p(w_1 \mid Spam) \times p(w_2 \mid Spam) \times p(w_3 \mid Spam) \times \ldots \times p(w_n \mid Spam)$$

given that each word is distinct. We then compare the two probabilities to find our most likely class. Applying this formula to the email "Lunch Money Money Money Money", we get:

$$
\begin{aligned}
p(Ham \mid "DearFriend") &= p(Dear \mid Ham) \times p(Friend \mid Ham) \\
&= 0.47 \times 0.29 \\
&= 0.136
\end{aligned}
$$

and

$$
\begin{aligned}
p(Spam \mid "DearFriend") &= p(Dear \mid Spam) \times p(Friend \mid Spam) \\
&= 0.29 \times 0.14 \\
&= 0.0406
\end{aligned}
$$

comparing the two, we see that $p(Ham) = 0.136 > p(Spam) = 0.0406$, so we predict the email is normal, and something you would want to see in your inbox.

3.3 Gaussian Naive Bayes

Accuracy: 70%

This is pretty much exactly the same method as Multinomal Naive Bayes, but instead of making a histogram of the words, we find the mean and standard deviation of how frequently the words occur and plot the normal curve of what that would look like, and multiply the likelihood of getting each word given spam or ham together to get a probability. We use this formula for likelihood:

$$L(\mu, \sigma \mid x) = \frac{1}{\sqrt{2}\pi\sigma^2} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ is the standard deviation.

We compare the probabilities for getting spam or ham, and pick whichever one is largest. Sometimes the value of the likelihood is so small and close to zero such that the computer can't keep track of the number, and we get something called underflow and errors occur. To combat this, we take the log of the probabilities and add everything together instead (because log transforms the multiplication into addition). We end up with negative numbers in this case. Any log will do but the natural log is most commonly used.

It is also important to note that we need more than one data point for a good estimate of the mean and standard deviation. If there is a word that appears only once in your dataset like "corn", we can't use this word to classify a given email. With only 1 data point, we can't get a good value for standard deviation or mean. If corn were to appear twice in an email, the math does not work out and everything gets ruined (because the probability would turn into 0), so it's important to exclude these types of words from your dataset. These types of words are also called stopwords, or words that stop your model from working. It's also important that you have enough data in your dataset to provide the model with enough data to work with to be reliable.