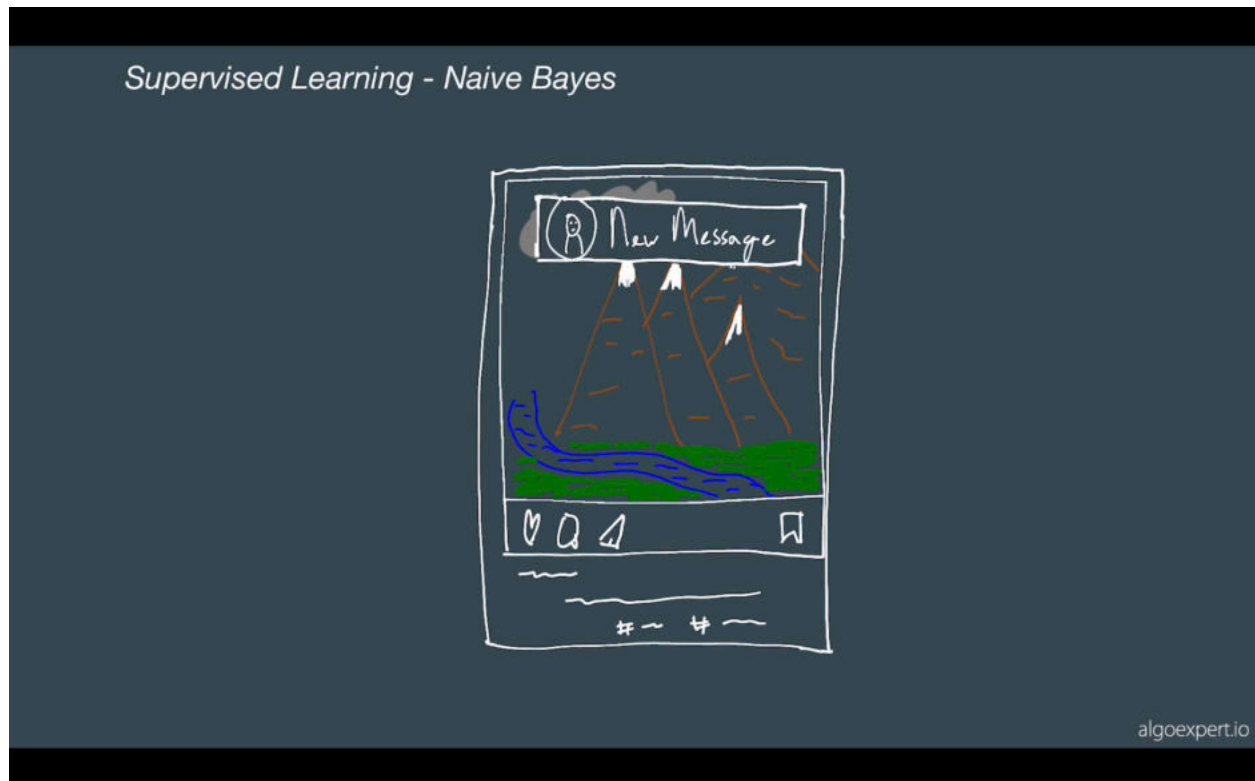
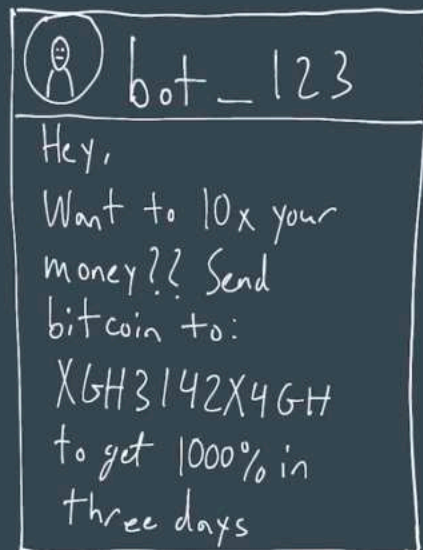


Naive Bayes



Welcome back everybody. This is ML Experts machine learning crash course. In this episode, we're going to be going over a model called the **Naive Bayes Classifier**. Let's say that as you're watching this video, you're also scrolling through your **Instagram feed**. Suddenly, someone sends you a direct message and do you decide to click on it. You proceed to read the message, from bot_123. Hey, want to 10x your money? Send Bitcoin to this address to get 1000% in three days.

Supervised Learning - Naive Bayes

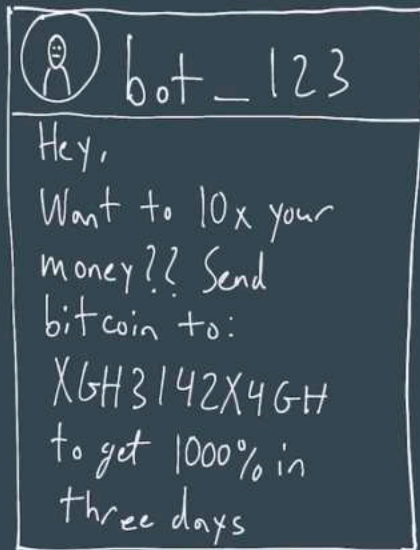


You're probably skeptical, and so am I. My suspicions are one, I don't recognize the sender, this person here. They're talking about some sort of quick returns. They mentioned Bitcoin and how they want me to send them some, and finally they don't use my name. But whatever my suspicions are, I'm not going to or reply to this message and now I've been needlessly distracted from my Instagram feed.

What would have been really nice is if Instagram could have detected that this message was probably suspicious and set the message aside, not send me a push notification and let me read the message at my own convenience later. If I were going to implement this, I would create some sort of filter rules.

These **filter rules** would indicate whether or not I should be sent a push notification or if the message should just go in my inbox and not necessarily alert me right away about it. The first rule would be if I'm not following them and they send me a message, then filter it. Likewise, if they're not following me. And finally, if the first message that they send to me doesn't contain my name, then also filter. Now, I'd apply these rules for about a month or so and see if the needless distractions decreased in some way. It's worth noting here that these rules aren't perfect.

Supervised Learning - Naive Bayes



bot_123
Hey,
Want to 10x your
money?? Send
bitcoin to:
XG3142X4GH
to get 1000% in
three days

Suspicious

- Don't recognize sender
- Talking about returns
- Bitcoin
- They don't use my name

Filter Rules

- I'm not following them
- They're not following me
- First message doesn't contain my name

algoexpert.io

Messages that are relevant to me could be filtered out as well. Messages that are spam or suspicious could be sent as a push notification anyway. What these rules effectively do is they create a function and this function takes in a particular message and it maps that message to a value of filter or don't filter.

Let's see if we can replace our filter rule function with some **machine learning model**. Now, these filter rules are actually a **heuristic** in the sense that they trade off something optimal for something simple and quick. Now, one idea is that we can apply filter rules to the contents of the particular message. For instance, **we could have filtered here based on the word Bitcoin**. Let's look more into that. Here, we have six messages and these were all detected to be spam and what we've done is we've gone through each of these spam messages and collected common words into a filter word list. Then we can add a filter rule which says if a particular message contains two or more of the words on this filter list, then go ahead and filter that message out. Now, this filter list and this particular rule of the two words could work for some messages. **But what if these words can be found in legitimate and spam messages?**

Hey, I found your wallet



Send money to my Apple wallet



For instance, I'd really like to be notified if someone reached out to me and said, hey **I found your wallet**. I wouldn't like to be notified if a bot reached out and said, **send money to my apple wallet**. Here, both messages contain the word **wallet**. So what we would really like from our model is to take in a message and produce the chance of the particular message being spam.

Chance in our case is equivalent to probability. And in probability notation, we would write P of spam given some message. This is a conditional probability statement. All it's saying is what's the probability of spam given some particular message.

$$P(\text{spam} | \text{message})$$

Now, ease of reading, we're going to replace message with **vector w**. Here, vector w is just list of words. So how do we actually figure this out? Well, there is a formula available called **Bayes theorem**.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{P(\text{spam}) \cdot P(\vec{w} | \text{spam})}{P(\text{spam}) \cdot P(\vec{w} | \text{spam}) + P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}$$

So what does this actually mean? Let's pretend for a moment that there's just a numerator. As well, let's get rid of these probabilities.

$$(spam | \vec{w}) = (spam) \cdot (\vec{w} | spam)$$

Now, starting with the left hand side here, consider what absolutely must be true for us to get a spam message given some particular words. It sounds obvious, but for one, **we have to get a spam message, and two the exact words in question and no other words have to appear in that spam message.** If both of these things are true, then we did fulfill the left-hand side and we did get a spam message with those particular words in it.

Well, **not every message that we receive is going to be spam**, but some fraction are, say 10%. As well, for every spam message that we do receive, **they're not going to contain the exact word in question but some will**, say 5%. These percentages that we're mentioning are just probabilities. And since we're looking for **the probability that we got spam given some particular words**, we have to add probabilities on the right hand side. Note that and here is just multiplication in terms of probability. Okay, so I think we get the idea.

$$P(\text{spam} | \vec{w}) = P(\text{spam}) \cdot P(\vec{w} | \text{spam})$$

So why is the denominator there? In general, **we could've gotten a spam message with those particular words in question or we could've gotten a legitimate message or not spam given those same words.** Both could have happened because we're considering these probabilities. **The denominator takes both cases into account.** The numerator is the same as the left-hand term here on the denominator, because that's what we want to know. Out of everything that could have happened (both cases on the denominator), what fraction of those things (the numerator) turned out to be spam?

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{P(\text{spam}) \cdot P(\vec{w} | \text{spam})}{P(\text{spam}) \cdot P(\vec{w} | \text{spam}) + P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}$$

algoexpert.io

If we wanted to know the probability of a particular message being not spam, then we would simply put this term (second element of the denominator) over here in the numerator instead.

Supervised Learning - Naive Bayes

$$P(\text{not spam} | \vec{w}) = \frac{P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}{P(\text{spam}) \cdot P(\vec{w} | \text{spam}) + P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}$$

But for our case, we want to know the probability of spam. So we'll keep this term in the numerator. Note that the equivalent of or in probability is just addition.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{P(\text{spam}) \cdot P(\vec{w} | \text{spam})}{P(\text{spam}) \cdot P(\vec{w} | \text{spam}) + P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}$$
$$\vec{w} = [\text{'wallet'}]$$

09:46 algoexpert.io











We haven't looked at exactly how this works with an example, so let's do that now. To start off, let's consider the one word message **wallet**. **Remember, that the goal is to get the probability that a particular message is spam given that that message contains particular words.** In our case, these words is just one word and that word is wallet. In order to calculate this, $P(\text{spam} | w)$, we have to know what each of these terms in the numerator and denominator are equal to.

$P(\text{spam})$ is just the probability that a particular message is spam in general, regardless of the words it contains. This is why we don't see any term w in here: $P(\text{spam})$. Imagine if all of the messages that we've seen so far have been spam. Well, then the probability of spam would be 100%. Likewise, if every message that we've seen so far has not been spam, then the probability of spam would be 0%.

Let's say for our case that we've seen two spam messages out of 10. This means that our probability of spam is 20%.

Supervised Learning - Naive Bayes

$$P(\text{spam}) = 20\%$$

Spam	Not Spam (legit)
 bot_101	 bot_106
 bot_102	 bot_107
	 bot_103
	 bot_108
	 bot_104
	 bot_109
	 bot_105
	 bot_110

algoexpert.io

Let's go ahead and plug that in. Here was P of spam, here was P of spam and here was P of not spam. Well, since we know the P of spam is 20% and every message is either going to be spam or not spam, then to get the probability of not spam, we just take one minus 20% which is 80%.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{.20 \cdot P(\vec{w} | \text{spam})}{.20 \cdot P(\vec{w} | \text{spam}) + .80 \cdot P(\vec{w} | \text{not spam})}$$

$\vec{w} = [\text{'wallet'}]$

algoexpert.io

The next term, (P of w / given spam) represents in our case, **the probability that our one word message 'wallet', appears in a spam message**. This means that **wallet** appeared in the message and **also not any other word**. This is the same way as saying, we received a **one word message with the word wallet in it**.

But to actually calculate this, we need to figure out how to represent not any other word appearing in spam messages. We can represent not any other word by defining a vocabulary. A **vocabulary** will be a list of words that our model understands or recognizes. Now we can define w , or our message wallet, in terms of our vocabulary.

Supervised Learning - Naive Bayes

$$P(\vec{w} | \text{spam})$$

↓ expanded

$$P(\text{wallet, not any other word} | \text{spam})$$

algoexpert.io

So here, let's say that the word wallet is the 47th word in our vocabulary, and we can represent not any other word as not the zero (*not v0*) with word in our vocabulary. Not all of these words in our vocabulary and finally not the last or the 99th (*not v47*) word in our vocabulary. So all this is saying is that the only word in our message is the 47th word in our vocabulary. All of the other words in our vocabulary are not in our message.

$$P(\vec{w} | \text{spam})$$

↓ expanded

$$P(\text{wallet, not any other word} | \text{spam})$$

↓ apply vocab

$$P(\vec{v}_{47}, \text{not } \vec{v}_0, \dots, \text{not } \vec{v}_{99} | \text{spam})$$

$$\text{vocab} = \vec{V} = [\text{'hi'}, \text{'hello'}, \text{'up'}, \text{'seattle'}, \dots, \text{'door'}]$$

Okay, so how do we calculate this then? Well, there's a formula we can use based on the **probability chain rule**.

Supervised Learning - Naive Bayes

$$P(\vec{v}_{47}, \text{not } \vec{v}_0, \dots, \text{not } \vec{v}_{99} \mid \text{spam}) =$$

To start, we'd have to calculate the probability of not seeing the last word in our vocabulary given some spam message ($P(\text{not } v_{99} \mid \text{spam})$). We then multiply that by not seeing the second to last word in our vocabulary given that we have already not seen the final word in our vocabulary in a spam message ($P(\text{not } v_{98} \mid \text{not } v_{99} \text{ spam})$). And then we would continue this pattern on with the third to the last word in our vocabulary. And this would continue for all of the words in our vocabulary until we reached the probability of seeing the 47th word given that we have not seen any other word in our vocabulary in a spam message.

Supervised Learning - Naive Bayes

$$\begin{aligned} P(\vec{v}_{47}, \text{not } \vec{v}_0, \dots, \text{not } \vec{v}_{99} | \text{spam}) = \\ P(\text{not } \vec{v}_{99} | \text{spam}) \\ \cdot P(\text{not } \vec{v}_{98} | \text{not } \vec{v}_{99}, \text{spam}) \\ \cdot P(\text{not } \vec{v}_{97} | \text{not } \vec{v}_{98}, \text{not } \vec{v}_{99}, \text{spam}) \\ \cdot \dots \\ \cdot P(\vec{v}_{47} | \text{not } \vec{v}_1, \dots, \text{not } \vec{v}_{99}, \text{spam}) \end{aligned}$$

Now, this is a lot to calculate. Imagine if our vocabulary had thousands of words in it. There would be tons of terms here and our model would likely become unusable because of the degree of calculation we'd have to perform. **An assumption we can make is that the presence or absence of a particular word does not depend on the presence or absence of any other word.** What this means is that we can now cancel out all of these terms within these calculations. Now, even though this does help us in terms of calculation, it does add **bias** to our model now.

Supervised Learning - Naive Bayes

$$\begin{aligned} P(\vec{v}_{47}, \text{not } \vec{v}_0, \dots, \text{not } \vec{v}_{99} | \text{spam}) = \\ P(\text{not } \vec{v}_{99} | \text{spam}) \\ \cdot P(\text{not } \vec{v}_{98} | \text{not } \vec{v}_{99}, \text{spam}) \\ \cdot P(\text{not } \vec{v}_{97} | \text{not } \vec{v}_{98}, \text{not } \vec{v}_{99}, \text{spam}) \\ \cdot P(\vec{v}_{47} | \text{not } \vec{v}_1, \dots, \text{not } \vec{v}_{99}, \text{spam}) \end{aligned}$$

algoexpert.io

This bias can hinder the model's ability to understand nuances within messages. For instance, the model will no longer account for an increased probability of seeing the word **London** upon seeing the word **England**. Now, even though these words are likely to appear together, **the simplifying assumption enforces that the model see these two words in isolation.** This trade-off buys us the ability to still use this model given an extremely large vocabulary, and even though we have this **independence assumption**, it tends to work well in practice. We said earlier that this model was based on Bayes theorem or Bayes rule. Now that we have this simplifying assumption here, our model represents what's called a **Naive Bayes Model**. So now our formula is just equal to this.

Supervised Learning - Naive Bayes

$$P(\vec{v}_{47}, \text{not } \vec{v}_0, \dots, \text{not } \vec{v}_{99} | \text{spam}) =$$

London England

Naive Bayes

$$= P(\vec{v}_{47} | \text{spam}) \cdot P(\text{not } \vec{v}_0 | \text{spam}) \dots \\ \cdot P(\text{not } \vec{v}_{99} | \text{spam})$$

What we're doing now is simply multiplying the probabilities of seeing particular words in a spam message by treating single word independently from one another as if they're not even in the same message. So now let's substitute this in for this term $P(w | \text{spam})$ in all our model.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{.20 \cdot P(\vec{w} | \text{spam})}{.20 \cdot P(\vec{w} | \text{spam}) + .80 \cdot P(\vec{w} | \text{not spam})}$$

$\vec{w} = [\text{'wallet'}]$

algoexpert.io

One thing you may have noticed here is this little symbol. All that means is not. So here this means not spam and this means not the last word in our vocabulary.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{.20 \cdot P(\vec{v}_{y7} | \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \text{spam})}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{y7} | \neg \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \neg \text{spam})}$$
$$\vec{w} = [\text{'wallet'}]$$

algoexpert.io

As a recap, all we've done is take this term and substitute the term $P(w | \text{spam})$ based on our **naive independence assumption**.

Supervised Learning - Naive Bayes

$$\begin{aligned} &P(\vec{w} \mid \text{spam}) \\ &\quad \downarrow \text{expanded} \\ &P(\text{wallet, not any other word} \mid \text{spam}) \\ &\quad \downarrow \text{apply vocab} \\ &P(\vec{v}_{47}, \text{not } \vec{v}_0, \dots, \text{not } \vec{v}_{99} \mid \text{spam}) \\ &\quad \downarrow \text{naive independence} \\ &P(\vec{v}_{47} \mid \text{spam}) \cdot P(\text{not } \vec{v}_0 \mid \text{spam}) \cdot \dots \cdot P(\text{not } \vec{v}_{99} \mid \text{spam}) \end{aligned}$$

algoexpert.io

So how do we actually calculate these terms (the term we've recently replaced)? Well, let's start with our first term here ($P(v_{47} \mid \text{spam})$). This term is just the probability that the 47th word in our vocabulary, wallet, appears in some spam message. To calculate this, we count the total number of spam messages which contain the word wallet and then divide by the total number of spam messages.

For instance, let's say that this message does contain the word wallet. That's indicated here with this check mark.

So, our probability would be one over two total messages. If both of these spam messages contained the word wallet, then the probability that the word wallet appears in a spam message would be 100%. For our particular example, the probability is actually 50%. This is because half of our spam messages contain the word wallet. So let's plug in 50% for this term.

Supervised Learning - Naive Bayes











$$P(\vec{v}_{47} = \text{'wallet'} \mid \text{spam}) = 50\%$$

Spam	Not Spam (legit)
<div>bot_101 ✓</div>	<div>bot_106</div>
<div>bot_102</div>	<div>bot_107</div>
	<div>bot_103</div>
	<div>bot_101</div>
	<div>bot_104</div>
	<div>bot_109</div>
	<div>bot_105</div>
	<div>bot_110</div>

The next term we're going to calculate is the probability that the final word in our vocabulary does not appear in a spam message. So let's calculate that now. We have the probability that the 99th term in our vocabulary, door, does not appear in a spam message ($P(-v_{99} \mid \text{spam})$). All that we do is count the number of spam messages which don't contain the word door. Here, both spam messages don't contain the word door so we have a 100% chance of the word door not appearing in a spam message. So let's go ahead and plug in 100% for this term.

Supervised Learning - Naive Bayes

$$P(\neg \vec{v}_{44} = \text{'door'} \mid \text{spam}) = 100\%$$

Spam	Not Spam (legit)
 bot_101 X	 bot_106
 bot_102 X	 bot_107
	 bot_103
	 bot_108
	 bot_104
	 bot_109
	 bot_105
	 bot_110

algoexpert.io

We'd continue that same process for every other word in our vocabulary as well. Here, since this term (the whole numerator) and this term on the left (of the denominator) are equivalent, we've just gone ahead and put this down here (the left side of denominator we mention).

On the right hand side of the addition, we need to calculate this term next $P(v_{47} \mid \neg \text{spam})$.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{.20 \cdot .50 \cdot \dots \cdot 1.0}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{q1} | \neg \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \neg \text{spam})}$$
$$\vec{w} = [\text{'wallet'}]$$

algoexpert.io

So now we'll be calculating the probability that the word wallet appears in a non spam or legitimate message. We calculate this by just counting the number of non spam messages which contain the word wallet. We then divide by the total number of non spam messages. Here, this would be 25% because two, out of eight total messages contain the word wallet. We can now plug in 25% for this term.

Supervised Learning - Naive Bayes

$$P(\vec{v}_{47} = \text{'wallet'} \mid \neg \text{spam}) = 25\%$$

Spam	Not Spam (legit)
<div>bot_101</div>	<div>bot_106</div> ✓
<div>bot_102</div>	<div>bot_107</div> ✓
	<div>bot_103</div>
	<div>bot_101</div>
	<div>bot_104</div>
	<div>bot_109</div>
	<div>bot_105</div>
	<div>bot_110</div>











algoexpert.io

Now we would continue this process as well for every other message in our vocabulary, but together we'll solve this final term here ($P(\neg v_{99} \mid \neg \text{spam})$). So now we want the probability that the 99th word in our vocabulary, door, does not appear in a legitimate message.

To calculate this, we simply count the number of legitimate messages which do not contain the word door. We then divide by the total number of legitimate messages. Here, since five out of eight total legitimate messages don't contain the word door, the probability is 62.5%. Let's plug in 62.5 for this term now.

Supervised Learning - Naive Bayes

$$P(\neg \vec{v}_{qq} = \text{'door'} \mid \neg \text{Spam}) = 62.5\%$$

Spam	Not Spam (legit)
<div> bot_101</div>	<div> bot_106</div>
<div> bot_102</div>	<div> bot_107 ✗</div>
	<div>✗  bot_103</div>
	<div>✗  bot_104</div>
	<div>✗  bot_105</div>
	<div> bot_108</div>
	<div> bot_109 ✗</div>
	<div> bot_110</div>

algoexpert.io

So now, assuming that we've repeated this process for every other word in our vocabulary, we can solve for this probability here.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) =$$

$$\frac{.20 \cdot .50 \cdot \dots \cdot 1.0}{.20 \cdot \dots + .80 \cdot .25 \cdot \dots \cdot .625}$$

$$\vec{w} = [\text{'wallet'}]$$

algoexpert.io

It turns out that the probability of spam given the exact message, wallet, is 47.7%. I think it makes sense that we're kind of in the middle between spam and not spam. If this were very close to zero, it would imply that there's a very low chance that this word is spam. If the probability was near 100, it would imply a very strong possibility that this one word message, wallet, is spam.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) =$$

47.7%

$$\vec{w} = [\text{'wallet'}]$$



word	spam	count
wallet	1	1

So let's bring up our model again. And what happens if our message contains more than just a single word?

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) =$$

$$\frac{.20 \cdot P(\vec{v}_{y1} | \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \text{spam})}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{y1} | \neg \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \neg \text{spam})}$$

$$\vec{w} = [\text{'wallet'}]$$

algosexpert.io

So for instance, if our message now contains **wallet and door**, everything stays the same except this term here ($P(v_{99} \mid \text{spam})$) no longer has a not and we would calculate it in the similar way that we calculated this ($P(v_{47} \mid \text{spam})$).

Supervised Learning - Naive Bayes

$$P(\text{spam} \mid \vec{w}) = \frac{.20 \cdot P(\vec{v}_{47} \mid \text{spam}) \cdot \dots \cdot P(\vec{v}_{99} \mid \text{spam})}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{47} \mid \neg \text{spam}) \cdot \dots \cdot P(\vec{v}_{99} \mid \neg \text{spam})}$$

$$\vec{w} = [\text{'wallet', 'door'}]$$

algoexpert.io

The little nots in front of the terms can actually be reduced down to a binary representation. This is called vectorizing and it's useful for when we're dealing with these models in some form software. So here, the 47th item in this 100 length array, is a one, because our input message does contain the word wallet. The same goes for the 99th term, because our message is now wallet door. Every other index in this array is zero because no other word appears in our message.

Supervised Learning - Naive Bayes

$$P(\text{spam} | \vec{w}) = \frac{.20 \cdot P(\vec{v}_{47} | \text{spam}) \cdot \dots \cdot P(\vec{v}_{99} | \text{spam})}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{47} | \neg \text{spam}) \cdot \dots \cdot P(\vec{v}_{99} | \neg \text{spam})}$$
$$\vec{w} = [0, 0, \dots, \overset{\vec{v}_{47}}{1}, \dots, \overset{\vec{v}_{99}}{0}, 1]$$

algoexpert.io

So let's go over a problem we can run into while using this model. Let's say for instance that this probability actually worked out to be 0%. What that means is that zero of our spam messages contain the word door.

Supervised Learning - Naive Bayes

$$P(\vec{v}_{qq} = \text{'door'} \mid \text{spam}) = 0\%$$

Spam	Not Spam (legit)
<div>bot_101</div>	<div>bot_106</div>
<div>bot_102</div>	<div>bot_107</div>
	<div>bot_103</div>
	<div>bot_108</div>
	<div>bot_104</div>
	<div>bot_109</div>
	<div>bot_105</div>
	<div>bot_110</div>

algoexpert.io

Well, in that case, if we put in the probability that the word door appears in the spam message, it will actually be zero. That's not good because what that means is the entire numerator here will be zero and no matter what other terms are here, the probability will work out to be zero since anything times zero is just zero. How do we get around this?

Supervised Learning - Naive Bayes











$$P(\text{spam} | \vec{w}) = \frac{.20 \cdot P(\vec{v}_{47} | \text{spam}) \cdot \dots \cdot 0}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{47} | \neg \text{spam}) \cdot \dots \cdot P(\vec{v}_{99} | \neg \text{spam})}$$
$$\vec{w} = [\text{'wallet', 'door'}]$$

algoexpert.io

Well, we can add something called Laplace smoothing. So here, zero out of two spam messages, contain the word door and that's where 0% probability came from. Laplace smoothing simply adds one to the numerator and two to the denominator. So here, the probability after **Laplace smoothing**, is actually 25%. Now Laplace smoothing **is applied to every word in the vocabulary**, not just the words that have zero probability.

Supervised Learning - Naive Bayes

$$P(\vec{v}_{qq} = \text{'door'} \mid \text{spam}) = 25\%$$

Spam	Not Spam (legit)
<div>  b_o_t_101 </div> <div>  b_o_t_102 </div>	<div>  b_o_t_106 </div> <div>  b_o_t_107 </div>
<div>  b_o_t_103 </div> <div>  b_o_t_104 </div> <div>  b_o_t_105 </div>	<div>  b_o_t_108 </div> <div>  b_o_t_109 </div> <div>  b_o_t_110 </div>

Laplace Smoothing

$$\frac{0+1}{2+2}$$

algoexpert.io

This is very commonly used in these types of models to avoid that zero numerator problem. So now that we've gone over the model, let's look at how we preprocess our data before using it and applying it for our model.

Supervised Learning - Naive Bayes

Hey, good point here — this is interesting.

algoexpert.io

Let's say that this is just one of the many messages that we have that we're considering in our model. The first thing we would do is remove the white space and the punctuation surrounding the words. So here, tokenization takes in our raw message, and produces a list or array of these separated words. Second, let's take out the words that don't add much information. This could be words like this, is or a. This is called stop word removal and it's quite common. Third, let's remove any non alphabetic characters from words. Here, this was a non alphabetic character, so we've just removed that entirely. Finally, there's something called **stemming**, which removes the ending like I-N-Gs or E-Ss with the intent on preserving the stem or the root of the word. Besides stemming, there's something called **lemmatization**.

Supervised Learning - Naive Bayes

Hey, good point here — this is interesting.

↓ tokenization

['Hey', 'good', 'point', 'here', '—', 'this', 'is', 'interesting']

↓ stop word removal

['Hey', 'good', 'point', '—', 'interesting']

↓ non-alphabetic removal

['Hey', 'good', 'point', 'interesting']

↓ stemming

['Hey', 'good', 'point', 'interest']

Stemming with a hard fast rule of simply removing these endings to form different stems, lemmatization takes a more nuanced approach and would map these two words to the same root word. The problem with this is that **lemmatization is often more expensive** than just stemming your raw messages. So if you have large amounts of data, you may want to go with stemming over lemmatization. In our case, we're just gonna go with stemming.

Supervised Learning - Naive Bayes

Stemming

Studying → Study
Studies → Studi

Lemmatization

Studying → Study
Studies → Study

algoexpert.io

As well, just be aware that if you do lowercase, all of your raw messages, you could lose the understanding of particular names or pronouns and they could resolve down into the regular noun forms.

Lowercasing

Mark(name) → mark(noun)

The entire process of going from a raw message to something that we can use in our naive based model, is called a **featurizing**. This is because we now have features for our model as opposed to the raw messages.

Supervised Learning - Naive Bayes

Featurizing

Hey, good point here — this is interesting.

↓ **tokenization**

['Hey', 'good', 'point', 'here', '—', 'this', 'is', 'interesting']

↓ **stop word removal**

['Hey', 'good', 'point', '—', 'interesting']

↓ **non-alphabetic removal**

['Hey', 'good', 'point', 'interesting']

↓ **stemming**

['Hey', 'good', 'point', 'interest']

algoexpert.io

And as we talked about earlier, we can take this featurized input and we can represent it in terms of our vocabulary. This is called **vectorizing**. Now, this vectorization will always be binary in the case of the model that we're using.

Supervised Learning - Naive Bayes

`['Hey', 'good', 'point', 'interest']`
↓ *vectorizing*
`[0, 0, 1, 0, ..., 0, 0, 1, 0]`
binary

algoexpert.io

For some general terms of our model, we have the priors, this is the prior of spam and the prior of not spam.

Priors

$$\frac{P(\text{spam}) \cdot P(\vec{w} | \text{spam})}{P(\text{spam}) \cdot P(\vec{w} | \text{spam}) + P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}$$

These are referenced as the likelihoods. So this would be the spam likelihoods and these would be the legitimate or non spam likelihoods.

Likelihoods

$$\frac{P(\text{spam}) \cdot \overbrace{P(\vec{w} | \text{spam})}}{\underbrace{P(\text{spam}) \cdot \underbrace{P(\vec{w} | \text{spam})} + P(\text{not spam}) \cdot \underbrace{P(\vec{w} | \text{not spam})}}}$$

Bernoulli:

Now we've chosen in our current model to represent these likelihoods as the presence or absence of words and this is called the **Bernoulli** model. We'll go over other types of models that we can use later.

Finally, if you get asked, this is called the evidence and what you're solving for, the result of this entire model is called the posterior.

Evidence

$$\frac{P(\text{spam}) \cdot P(\vec{w} | \text{spam})}{P(\text{spam}) \cdot P(\vec{w} | \text{spam}) + P(\text{not spam}) \cdot P(\vec{w} | \text{not spam})}$$

All right, so let's go over what a fully trained model would look like. Well, we would need the prior of spam and the prior of legitimate messages. We would also need a spam likelihood map and a legitimate likelihood map. All this means is that the key would be the particular word in our vocabulary and the value would either be the spam likelihood in this map or the legitimate likelihood in this map.

Supervised Learning - Naive Bayes

$$\frac{.20 \cdot P(\vec{v}_{y7} | \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \text{spam})}{.20 \cdot \dots + .80 \cdot P(\vec{v}_{y7} | \neg \text{spam}) \cdot \dots \cdot P(\neg \vec{v}_{q9} | \neg \text{spam})}$$

prior_spam

spam_likelihood

prior_legit

legit_likelihood

algoexpert.io

But the way that we calculate these, are the same ways that we've reviewed before. Here, we would use the legitimate likelihood and here we would just take one minus the legitimate likelihood since it does not appear. All right, well that wraps up this video. Thanks for joining, join us in our next video as we continue.