

# 4

## Probabilistic Learning – Classification Using Naive Bayes

When a meteorologist provides a weather forecast, precipitation is typically described with terms such as "70 percent chance of rain." Such forecasts are known as probability of precipitation reports. Have you ever considered how they are calculated? It is a puzzling question, because in reality, either it will rain or not.

Weather estimates are based on probabilistic methods or those concerned with describing uncertainty. They use data on past events to extrapolate future events. In the case of weather, the chance of rain describes the proportion of prior days to similar measurable atmospheric conditions in which precipitation occurred. A 70 percent chance of rain implies that in 7 out of the 10 past cases with similar conditions, precipitation occurred somewhere in the area.

This chapter covers the Naive Bayes algorithm, which uses probabilities in much the same way as a weather forecast. While studying this method, you will learn:

- Basic principles of probability
- The specialized methods and data structures needed to analyze text data with R
- How to employ Naive Bayes to build an SMS junk message filter

If you've taken a statistics class before, some of the material in this chapter may be a review. Even so, it may be helpful to refresh your knowledge on probability, as these principles are the basis of how Naive Bayes got such a strange name.

## Understanding Naive Bayes

The basic statistical ideas necessary to understand the Naive Bayes algorithm have existed for centuries. The technique descended from the work of the 18<sup>th</sup> century mathematician Thomas Bayes, who developed foundational principles to describe the probability of events, and how probabilities should be revised in the light of additional information. These principles formed the foundation for what are now known as **Bayesian methods**.

We will cover these methods in greater detail later on. But, for now, it suffices to say that a probability is a number between 0 and 1 (that is, between 0 percent and 100 percent), which captures the chance that an event will occur in the light of the available evidence. The lower the probability, the less likely the event is to occur. A probability of 0 indicates that the event will definitely not occur, while a probability of 1 indicates that the event will occur with 100 percent certainty.

Classifiers based on Bayesian methods utilize training data to calculate an observed probability of each outcome based on the evidence provided by feature values. When the classifier is later applied to unlabeled data, it uses the observed probabilities to predict the most likely class for the new features. It's a simple idea, but it results in a method that often has results on par with more sophisticated algorithms. In fact, Bayesian classifiers have been used for:

- Text classification, such as junk e-mail (spam) filtering
- Intrusion or anomaly detection in computer networks
- Diagnosing medical conditions given a set of observed symptoms

Typically, Bayesian classifiers are best applied to problems in which the information from numerous attributes should be considered simultaneously in order to estimate the overall probability of an outcome. While many machine learning algorithms ignore features that have weak effects, Bayesian methods utilize all the available evidence to subtly change the predictions. If large number of features have relatively minor effects, taken together, their combined impact could be quite large.

## Basic concepts of Bayesian methods

Before jumping into the Naive Bayes algorithm, it's worth spending some time defining the concepts that are used across Bayesian methods. Summarized in a single sentence, Bayesian probability theory is rooted in the idea that the estimated likelihood of an **event**, or a potential outcome, should be based on the evidence at hand across multiple **trials**, or opportunities for the event to occur.

The following table illustrates events and trials for several real-world outcomes:

Event	Trial
Heads result	Coin flip
Rainy weather	A single day
Message is spam	Incoming e-mail message
Candidate becomes president	Presidential election
Win the lottery	Lottery ticket

Bayesian methods provide insights into how the probability of these events can be estimated from the observed data. To see how, we'll need to formalize our understanding of probability.

## Understanding probability

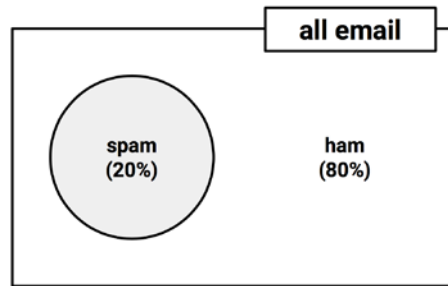
The probability of an event is estimated from the observed data by dividing the number of trials in which the event occurred by the total number of trials. For instance, if it rained 3 out of 10 days with similar conditions as today, the probability of rain today can be estimated as  $3/10 = 0.30$  or 30 percent. Similarly, if 10 out of 50 prior email messages were spam, then the probability of any incoming message being spam can be estimated as  $10/50 = 0.20$  or 20 percent.

To denote these probabilities, we use notation in the form  $P(A)$ , which signifies the probability of event  $A$ . For example,  $P(\text{rain}) = 0.30$  and  $P(\text{spam}) = 0.20$ .

The probability of all the possible outcomes of a trial must always sum to 1, because a trial always results in some outcome happening. Thus, if the trial has two outcomes that cannot occur simultaneously, such as rainy versus sunny or spam versus ham (nospam), then knowing the probability of either outcome reveals the probability of the other. For example, given the value  $P(\text{spam}) = 0.20$ , we can calculate  $P(\text{ham}) = 1 - 0.20 = 0.80$ . This concludes that spam and ham are **mutually exclusive and exhaustive** events, which implies that they cannot occur at the same time and are the only possible outcomes.

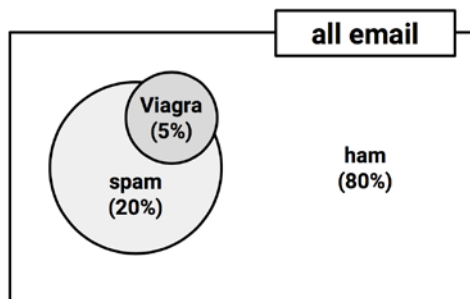
Because an event cannot simultaneously happen and not happen, an event is always mutually exclusive and exhaustive with its **complement**, or the event comprising of the outcomes in which the event of interest does not happen. The complement of event  $A$  is typically denoted  $A^c$  or  $A'$ . Additionally, the shorthand notation  $P(\neg A)$  can be used to denote the probability of event  $A$  not occurring, as in  $P(\neg \text{spam}) = 0.80$ . This notation is equivalent to  $P(A^c)$ .

To illustrate events and their complements, it is often helpful to imagine a two-dimensional space that is partitioned into probabilities for each event. In the following diagram, the rectangle represents the possible outcomes for an e-mail message. The circle represents the 20 percent probability that the message is spam. The remaining 80 percent represents the complement  $P(\neg spam)$  or the messages that are not spam:



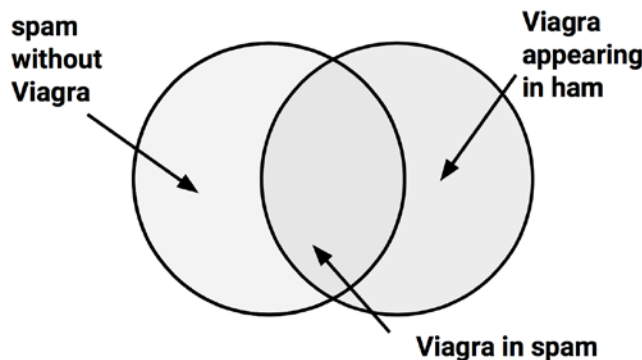
## Understanding joint probability

Often, we are interested in monitoring several nonmutually exclusive events for the same trial. If certain events occur with the event of interest, we may be able to use them to make predictions. Consider, for instance, a second event based on the outcome that an e-mail message contains the word Viagra. In most cases, this word is likely to appear only in a spam message; its presence in an incoming e-mail is therefore a very strong piece of evidence that the message is spam. The preceding diagram, updated for this second event, might appear as shown in the following diagram:



Notice in the diagram that the Viagra circle does not completely fill the spam circle, nor is it completely contained by the spam circle. This implies that not all spam messages contain the word Viagra and not every e-mail with the word Viagra is spam.

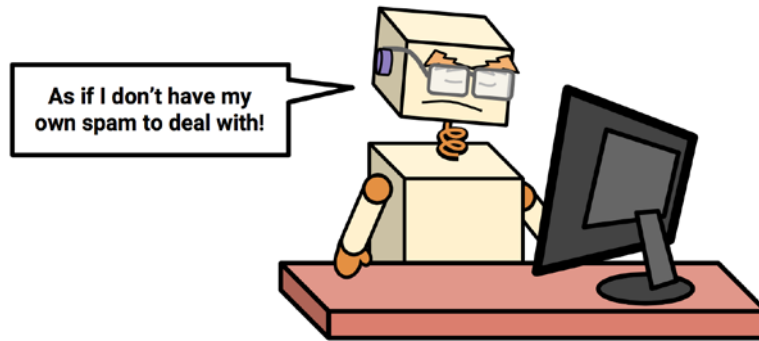
To zoom in for a closer look at the overlap between the spam and Viagra circles, we'll employ a visualization known as a **Venn diagram**. First used in the late 19<sup>th</sup> century by John Venn, the diagram uses circles to illustrate the overlap between sets of items. In most Venn diagrams, the size of the circles and the degree of the overlap is not meaningful. Instead, it is used as a reminder to allocate probability to all possible combinations of events:



We know that 20 percent of all messages were spam (the left circle) and 5 percent of all messages contained the word Viagra (the right circle). We would like to quantify the degree of overlap between these two proportions. In other words, we hope to estimate the probability that both  $P(\text{spam})$  and  $P(\text{Viagra})$  occur, which can be written as  $P(\text{spam} \cap \text{Viagra})$ . The upside down 'U' symbol signifies the **intersection** of the two events; the notation  $A \cap B$  refers to the event in which both  $A$  and  $B$  occur.

Calculating  $P(\text{spam} \cap \text{Viagra})$  depends on the **joint probability** of the two events or how the probability of one event is related to the probability of the other. If the two events are totally unrelated, they are called **independent events**. This is not to say that independent events cannot occur at the same time; event independence simply implies that knowing the outcome of one event does not provide any information about the outcome of the other. For instance, the outcome of a heads result on a coin flip is independent from whether the weather is rainy or sunny on any given day.

If all events were independent, it would be impossible to predict one event by observing another. In other words, **dependent events** are the basis of predictive modeling. Just as the presence of clouds is predictive of a rainy day, the appearance of the word Viagra is predictive of a spam e-mail.



Calculating the probability of dependent events is a bit more complex than for independent events. If  $P(\text{spam})$  and  $P(\text{Viagra})$  were independent, we could easily calculate  $P(\text{spam} \cap \text{Viagra})$ , the probability of both events happening at the same time. Because 20 percent of all the messages are spam, and 5 percent of all the e-mails contain the word Viagra, we could assume that 1 percent of all messages are spam with the term Viagra. This is because  $0.05 * 0.20 = 0.01$ . More generally, for independent events  $A$  and  $B$ , the probability of both happening can be expressed as  $P(A \cap B) = P(A) * P(B)$ .

This said, we know that  $P(\text{spam})$  and  $P(\text{Viagra})$  are likely to be highly dependent, which means that this calculation is incorrect. To obtain a reasonable estimate, we need to use a more careful formulation of the relationship between these two events, which is based on advanced Bayesian methods.

## Computing conditional probability with Bayes' theorem

The relationships between dependent events can be described using **Bayes' theorem**, as shown in the following formula. This formulation provides a way of thinking about how to revise an estimate of the probability of one event in light of the evidence provided by another event:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

The notation  $P(A | B)$  is read as the probability of event  $A$ , given that event  $B$  occurred. This is known as **conditional probability**, since the probability of  $A$  is dependent (that is, conditional) on what happened with event  $B$ . Bayes' theorem tells us that our estimate of  $P(A | B)$  should be based on  $P(A \cap B)$ , a measure of how often  $A$  and  $B$  are observed to occur together, and  $P(B)$ , a measure of how often  $B$  is observed to occur in general.

Bayes' theorem states that the best estimate of  $P(A | B)$  is the proportion of trials in which  $A$  occurred with  $B$  out of all the trials in which  $B$  occurred. In plain language, this tells us that if we know event  $B$  occurred, the probability of event  $A$  is higher the more often that  $A$  and  $B$  occur together each time  $B$  is observed. In a way, this adjusts  $P(A \cap B)$  for the probability of  $B$  occurring; if  $B$  is extremely rare,  $P(B)$  and  $P(A \cap B)$  will always be small; however, if  $A$  and  $B$  almost always happen together,  $P(A | B)$  will be high regardless of the probability of  $B$ .

By definition,  $P(A \cap B) = P(A | B) * P(B)$ , a fact that can be easily derived by applying a bit of algebra to the previous formula. Rearranging this formula once more with the knowledge that  $P(A \cap B) = P(B \cap A)$  results in the conclusion that  $P(A \cap B) = P(B | A) * P(A)$ , which we can then use in the following formulation of Bayes' theorem:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

In fact, this is the traditional way in which Bayes' theorem has been specified, for reasons that will become clear as we apply it to machine learning. First, to better understand how Bayes' theorem works in practice, let's revisit our hypothetical spam filter.

Without knowledge of an incoming message's content, the best estimate of its spam status would be  $P(spam)$ , the probability that any prior message was spam, which we calculated previously to be 20 percent. This estimate is known as the **prior probability**.

Suppose that you obtained additional evidence by looking more carefully at the set of previously received messages to examine the frequency that the term Viagra appeared. The probability that the word Viagra was used in previous spam messages, or  $P(Viagra | spam)$ , is called the **likelihood**. The probability that Viagra appeared in any message at all, or  $P(Viagra)$ , is known as the **marginal likelihood**.

By applying Bayes' theorem to this evidence, we can compute a **posterior probability** that measures how likely the message is to be spam. If the posterior probability is greater than 50 percent, the message is more likely to be spam than ham and it should perhaps be filtered. The following formula shows how Bayes' theorem is applied to the evidence provided by the previous e-mail messages:

$$P(\text{spam}|\text{Viagra}) = \frac{P(\text{Viagra}|\text{spam})P(\text{spam})}{P(\text{Viagra})}$$

likelihood
prior probability

posterior probability
marginal likelihood

To calculate these components of Bayes' theorem, it helps to construct a **frequency table** (shown on the left in the following diagram) that records the number of times Viagra appeared in spam and ham messages. Just like a two-way cross-tabulation, one dimension of the table indicates levels of the class variable (spam or ham), while the other dimension indicates levels for features (Viagra: yes or no). The cells then indicate the number of instances having the particular combination of class value and feature value. The frequency table can then be used to construct a **likelihood table**, as shown on right in the following diagram. The rows of the likelihood table indicate the conditional probabilities for Viagra (yes/no), given that an e-mail was either spam or ham:

Frequency	Viagra		Total
	Yes	No	
spam	4	16	20
ham	1	79	80
Total	5	95	100

Likelihood	Viagra		Total
	Yes	No	
spam	4 / 20	16 / 20	20
ham	1 / 80	79 / 80	80
Total	5 / 100	95 / 100	100

The likelihood table reveals that  $P(\text{Viagra}=\text{Yes} | \text{spam}) = 4/20 = 0.20$ , indicating that the probability is 20 percent that a message contains the term Viagra, given that the message is spam. Additionally, since  $P(A \cap B) = P(B | A) * P(A)$ , we can calculate  $P(\text{spam} \cap \text{Viagra})$  as  $P(\text{Viagra} | \text{spam}) * P(\text{spam}) = (4/20) * (20/100) = 0.04$ . The same result can be found in the frequency table, which notes that 4 out of the 100 messages were spam with the term Viagra. Either way, this is four times greater than the previous estimate of 0.01 we calculated as  $P(A \cap B) = P(A) * P(B)$  under the false assumption of independence. This, of course, illustrates the importance of Bayes' theorem while calculating joint probability.



To compute the posterior probability,  $P(\text{spam} \mid \text{Viagra})$ , we simply take  $P(\text{Viagra} \mid \text{spam}) * P(\text{spam}) / P(\text{Viagra})$  or  $(4/20) * (20/100) / (5/100) = 0.80$ . Therefore, the probability is 80 percent that a message is spam, given that it contains the word Viagra. In light of this result, any message containing this term should probably be filtered.

This is very much how commercial spam filters work, although they consider a much larger number of words simultaneously while computing the frequency and likelihood tables. In the next section, we'll see how this concept is put to use when additional features are involved.

## The Naive Bayes algorithm

The **Naive Bayes** algorithm describes a simple method to apply Bayes' theorem to classification problems. Although it is not the only machine learning method that utilizes Bayesian methods, it is the most common one. This is particularly true for text classification, where it has become the de facto standard. The strengths and weaknesses of this algorithm are as follows:

Strengths	Weaknesses
<ul style="list-style-type: none"> <li>Simple, fast, and very effective</li> <li>Does well with noisy and missing data</li> <li>Requires relatively few examples for training, but also works well with very large numbers of examples</li> <li>Easy to obtain the estimated probability for a prediction</li> </ul>	<ul style="list-style-type: none"> <li>Relies on an often-faulty assumption of equally important and independent features</li> <li>Not ideal for datasets with many numeric features</li> <li>Estimated probabilities are less reliable than the predicted classes</li> </ul>

The Naive Bayes algorithm is named as such because it makes some "naive" assumptions about the data. In particular, Naive Bayes assumes that all of the features in the dataset are equally important and independent. These assumptions are rarely true in most real-world applications.

For example, if you were attempting to identify spam by monitoring e-mail messages, it is almost certainly true that some features will be more important than others. For example, the e-mail sender may be a more important indicator of spam than the message text. Additionally, the words in the message body are not independent from one another, since the appearance of some words is a very good indication that other words are also likely to appear. A message with the word Viagra will probably also contain the words prescription or drugs.

However, in most cases when these assumptions are violated, Naive Bayes still performs fairly well. This is true even in extreme circumstances where strong dependencies are found among the features. Due to the algorithm's versatility and accuracy across many types of conditions, Naive Bayes is often a strong first candidate for classification learning tasks.



The exact reason why Naive Bayes works well in spite of its faulty assumptions has been the subject of much speculation. One explanation is that it is not important to obtain a precise estimate of probability, so long as the predictions are accurate. For instance, if a spam filter correctly identifies spam, does it matter whether it was 51 percent or 99 percent confident in its prediction? For one discussion of this topic, refer to: Domingos P, Pazzani M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*. 1997; 29:103-130.

## Classification with Naive Bayes

Let's extend our spam filter by adding a few additional terms to be monitored in addition to the term Viagra: Money, Groceries, and Unsubscribe. The Naive Bayes learner is trained by constructing a likelihood table for the appearance of these four words (labeled  $W_1$ ,  $W_2$ ,  $W_3$ , and  $W_4$ ), as shown in the following diagram for 100 e-mails:

Likelihood	Viagra ( $W_1$ )		Money ( $W_2$ )		Groceries ( $W_3$ )		Unsubscribe ( $W_4$ )		Total
	Yes	No	Yes	No	Yes	No	Yes	No	
spam	4 / 20	16 / 20	10 / 20	10 / 20	0 / 20	20 / 20	12 / 20	8 / 20	20
ham	1 / 80	79 / 80	14 / 80	66 / 80	8 / 80	71 / 80	23 / 80	57 / 80	80
Total	5 / 100	95 / 100	24 / 100	76 / 100	8 / 100	91 / 100	35 / 100	65 / 100	100

As new messages are received, we need to calculate the posterior probability to determine whether they are more likely to be spam or ham, given the likelihood of the words found in the message text. For example, suppose that a message contains the terms Viagra and Unsubscribe, but does not contain either Money or Groceries.

Using Bayes' theorem, we can define the problem as shown in the following formula. It captures the probability that a message is spam, given that  $Viagra = Yes$ ,  $Money = No$ ,  $Groceries = No$ , and  $Unsubscribe = Yes$ :

$$P(\text{spam} | W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4 | \text{spam}) P(\text{spam})}{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)}$$

For a number of reasons, this formula is computationally difficult to solve. As additional features are added, tremendous amounts of memory are needed to store probabilities for all of the possible intersecting events; imagine the complexity of a Venn diagram for the events for four words, let alone for hundreds or more.

The work becomes much easier if we can exploit the fact that Naive Bayes assumes independence among events. Specifically, it assumes **class-conditional independence**, which means that events are independent so long as they are conditioned on the same class value. Assuming conditional independence allows us to simplify the formula using the probability rule for independent events, which states that  $P(A \cap B) = P(A) * P(B)$ . Because the denominator does not depend on the class (spam or ham), it is treated as a constant value and can be ignored for the time being. This means that the conditional probability of spam can be expressed as:

$$P(\text{spam}|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \propto P(W_1|\text{spam})P(\neg W_2|\text{spam})P(\neg W_3|\text{spam})P(W_4|\text{spam})P(\text{spam})$$

And the probability that the message is ham can be expressed as:

$$P(\text{ham}|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \propto P(W_1|\text{ham})P(\neg W_2|\text{ham})P(\neg W_3|\text{ham})P(W_4|\text{ham})P(\text{ham})$$

Note that the equals symbol has been replaced by the proportional-to symbol (similar to a sideways, open-ended '8') to indicate the fact that the denominator has been omitted.

Using the values in the likelihood table, we can start filling numbers in these equations. The overall likelihood of spam is then:

$$(4/20) * (10/20) * (20/20) * (12/20) * (20/100) = 0.012$$

While the likelihood of ham is:

$$(1/80) * (66/80) * (71/80) * (23/80) * (80/100) = 0.002$$

Because  $0.012/0.002 = 6$ , we can say that this message is six times more likely to be spam than ham. However, to convert these numbers into probabilities, we need to perform one last step to reintroduce the denominator that had been excluded. Essentially, we must rescale the likelihood of each outcome by dividing it by the total likelihood across all possible outcomes.

In this way, the probability of spam is equal to the likelihood that the message is spam divided by the likelihood that the message is either spam or ham:

$$0.012/(0.012 + 0.002) = 0.857$$

Similarly, the probability of ham is equal to the likelihood that the message is ham divided by the likelihood that the message is either spam or ham:

$$0.002 / (0.012 + 0.002) = 0.143$$

Given the pattern of words found in this message, we expect that the message is spam with 85.7 percent probability and ham with 14.3 percent probability. Because these are mutually exclusive and exhaustive events, the probabilities sum to 1.

The Naive Bayes classification algorithm we used in the preceding example can be summarized by the following formula. The probability of level  $L$  for class  $C$ , given the evidence provided by features  $F_1$  through  $F_n$ , is equal to the product of the probabilities of each piece of evidence conditioned on the class level, the prior probability of the class level, and a scaling factor  $1/Z$ , which converts the likelihood values into probabilities:

$$P(C_L | F_1, \dots, F_n) = \frac{1}{Z} p(C_L) \prod_{i=1}^n p(F_i | C_L)$$

Although this equation seems intimidating, as the prior example illustrated, the series of steps is fairly straightforward. Begin by building a frequency table, use this to build a likelihood table, and multiply the conditional probabilities according to the Naive Bayes' rule. Finally, divide by the total likelihood to transform each class likelihood into a probability. After attempting this calculation a few times by hand, it will become second nature.

## The Laplace estimator

Before we employ Naive Bayes in more complex problems, there are some nuances to consider. Suppose that we received another message, this time containing all four terms: Viagra, Groceries, Money, and Unsubscribe. Using the Naive Bayes algorithm as before, we can compute the likelihood of spam as:

$$(4/20) * (10/20) * (0/20) * (12/20) * (20/100) = 0$$

The likelihood of ham is:

$$(1/80) * (14/80) * (8/80) * (23/80) * (80/100) = 0.00005$$

Therefore, the probability of spam is:

$$0 / (0 + 0.00005) = 0$$

The probability of ham is:

$$0.00005 / (0 + 0.00005) = 1$$

These results suggest that the message is spam with 0 percent probability and ham with 100 percent probability. Does this prediction make sense? Probably not. The message contains several words usually associated with spam, including Viagra, which is rarely used in legitimate messages. It is therefore very likely that the message has been incorrectly classified.

This problem might arise if an event never occurs for one or more levels of the class. For instance, the term Groceries had never previously appeared in a spam message. Consequently,  $P(\text{spam} | \text{groceries}) = 0\%$ .

Because probabilities in the Naive Bayes formula are multiplied in a chain, this 0 percent value causes the posterior probability of spam to be zero, giving the word Groceries the ability to effectively nullify and overrule all of the other evidence. Even if the e-mail was otherwise overwhelmingly expected to be spam, the absence of the word Groceries in spam will always veto the other evidence and result in the probability of spam being zero.

A solution to this problem involves using something called the **Laplace estimator**, which is named after the French mathematician Pierre-Simon Laplace. The Laplace estimator essentially adds a small number to each of the counts in the frequency table, which ensures that each feature has a nonzero probability of occurring with each class. Typically, the Laplace estimator is set to 1, which ensures that each class-feature combination is found in the data at least once.



The Laplace estimator can be set to any value and does not necessarily even have to be the same for each of the features. If you were a devoted Bayesian, you could use a Laplace estimator to reflect a presumed prior probability of how the feature relates to the class. In practice, given a large enough training dataset, this step is unnecessary and the value of 1 is almost always used.

Let's see how this affects our prediction for this message. Using a Laplace value of 1, we add one to each numerator in the likelihood function. The total number of 1 values must also be added to each conditional probability denominator. The likelihood of spam is therefore:

$$(5/24) * (11/24) * (1/24) * (13/24) * (20/100) = 0.0004$$

The likelihood of ham is:

$$(2/84) * (15/84) * (9/84) * (24/84) * (80/100) = 0.0001$$

This means that the probability of spam is 80 percent, and the probability of ham is 20 percent, which is a more plausible result than the one obtained when the term Groceries alone determined the result.

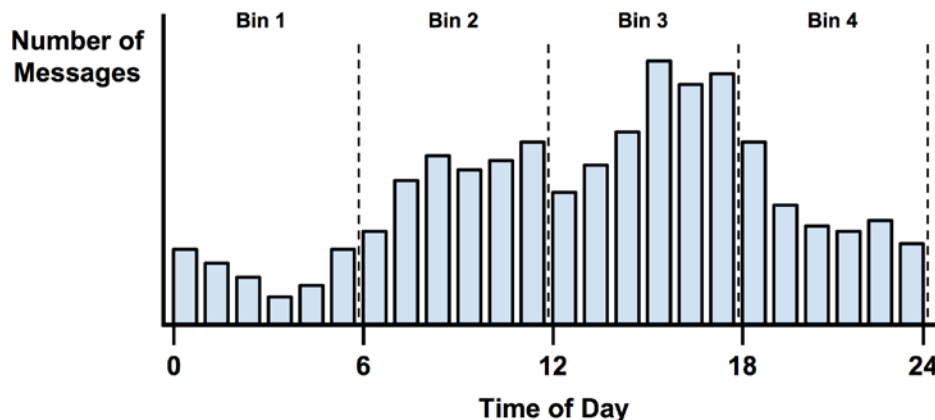
## Using numeric features with Naive Bayes

Because Naive Bayes uses frequency tables to learn the data, each feature must be categorical in order to create the combinations of class and feature values comprising of the matrix. Since numeric features do not have categories of values, the preceding algorithm does not work directly with numeric data. There are, however, ways that this can be addressed.


One easy and effective solution is to **discretize** numeric features, which simply means that the numbers are put into categories known as **bins**. For this reason, discretization is also sometimes called **binning**. This method is ideal when there are large amounts of training data, a common condition while working with Naive Bayes.

There are several different ways to discretize a numeric feature. Perhaps the most common is to explore the data for natural categories or **cut points** in the distribution of data. For example, suppose that you added a feature to the spam dataset that recorded the time of night or day the e-mail was sent, from 0 to 24 hours past midnight.

Depicted using a histogram, the time data might look something like the following diagram. In the early hours of the morning, the message frequency is low. The activity picks up during business hours and tapers off in the evening. This seems to create four natural bins of activity, as partitioned by the dashed lines indicating places where the numeric data are divided into levels of a new nominal feature, which could then be used with Naive Bayes:



Keep in mind that the choice of four bins was somewhat arbitrary based on the natural distribution of data and a hunch about how the proportion of spam might change throughout the day. We might expect that spammers operate in the late hours of the night or they may operate during the day, when people are likely to check their e-mail. This said, to capture these trends, we could have just as easily used three bins or twelve.

 If there are no obvious cut points, one option will be to discretize the feature using quantiles. You could divide the data into three bins with tertiles, four bins with quartiles, or five bins with quintiles.

One thing to keep in mind is that discretizing a numeric feature always results in a reduction of information as the feature's original granularity is reduced to a smaller number of categories. It is important to strike a balance here. Too few bins can result in important trends being obscured. Too many bins can result in small counts in the Naive Bayes frequency table, which can increase the algorithm's sensitivity to noisy data.

## Example – filtering mobile phone spam with the Naive Bayes algorithm

As the worldwide use of mobile phones has grown, a new avenue for electronic junk mail has opened for disreputable marketers. These advertisers utilize Short Message Service (SMS) text messages to target potential consumers with unwanted advertising known as SMS spam. This type of spam is particularly troublesome because, unlike e-mail spam, many cellular phone users pay a fee per SMS received. Developing a classification algorithm that could filter SMS spam would provide a useful tool for cellular phone providers.

Since Naive Bayes has been used successfully for e-mail spam filtering, it seems likely that it could also be applied to SMS spam. However, relative to e-mail spam, SMS spam poses additional challenges for automated filters. SMS messages are often limited to 160 characters, reducing the amount of text that can be used to identify whether a message is junk. The limit, combined with small mobile phone keyboards, has led many to adopt a form of SMS shorthand lingo, which further blurs the line between legitimate messages and spam. Let's see how a simple Naive Bayes classifier handles these challenges.