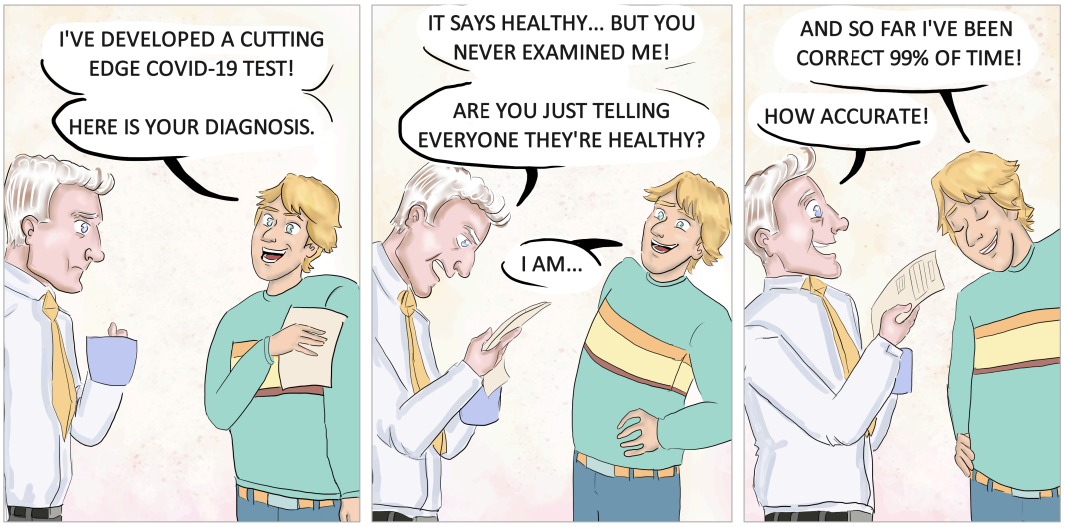
7 How do you measure classification models? Accuracy and its friends

56

published book

In this chapter

- types of errors a model can make: false positives and false negatives
- putting these errors in a table: the confusion matrix
- what are accuracy, recall, precision, F-score, sensitivity, and specificity, and how are they used to evaluate models
- what is the ROC curve, and how does it keep track of sensitivity and specificity at the same time



This chapter is slightly different from the previous two—it doesn't focus on building classification models; instead, it focuses on evaluating them. For a machine learning professional, being able to evaluate the performance of different models is as important as being able to train them. We seldom train a single model on a dataset; we train several different models and select the one that performs best. We also need to make sure models are of good quality before putting them in production. The quality of a model is not always trivial to measure, and in this chapter, we learn several techniques to evaluate our classification models. In chapter 4, we learned how to evaluate regression models, so we can think of this chapter as its analog but with classification models.

The simplest way to measure the performance of a classification model is by calculating its accuracy. However, we'll see that accuracy doesn't paint the whole picture, because some models exhibit great accuracy but are not good models anyway. To fix this, we'll define some useful metrics, such as precision and recall. Then we'll combine them into a new, more powerful metric called the F-score. These metrics are widely used by data scientists to evaluate their models. However, in other disciplines, such as medicine, other similar metrics are used, such as sensitivity and specificity. Using these last two metrics, we'll be able to build a curve called the receiver operating characteristic (ROC) curve. The ROC curve is a simple plot that gives us great insights into our models.

Accuracy: How often is my model correct?

In this section, we discuss accuracy, the simplest and most common measure of classification models. The accuracy of a model is the percentage of times that a model is correct. In other words, it is the ratio between the number of correctly predicted data points and the total number of data points. For example, if we evaluate a model on a test dataset of 1,000 samples, and the model predicted the correct label of the samples 875 times, then this model has an accuracy of 875/1000 = 0.875, or 87.5%.

Accuracy is the most common way to evaluate a classification model, and we should always use it. However, sometimes accuracy doesn't fully

describe the performance of the model, as we'll see shortly. Let's begin by looking at two examples that we'll study throughout this chapter.

TWO EXAMPLES OF MODELS: CORONAVIRUS AND SPAM EMAIL

In this chapter, we use our metrics to evaluate several models on two datasets. The first dataset is a medical dataset of patients, where some of them have been diagnosed with coronavirus. The second dataset is a dataset of emails that have been labeled as spam or not spam. As we learned in chapter 1, *spam* is the term used for junk email, and *ham* is the term used for email that is not spam. In chapter 8, we'll study a dataset like this in much more detail, when we learn the naive Bayes algorithm. In this chapter, we aren't building models. Instead, we use the models as black boxes and evaluate them based on how many of the data points they predict correctly or incorrectly. Both datasets are completely imaginary.

MEDICAL DATASET: A SET OF PATIENTS DIAGNOSED WITH CORONAVIRUS

Our first dataset is a medical dataset with 1,000 patients. Out of them, 10 have been diagnosed with coronavirus, and the remaining 990 have been diagnosed as healthy. Thus, the labels in this dataset are "sick" or "healthy," corresponding to the diagnosis. The goal of a model would be to predict the diagnosis based on the features of each patient.

EMAIL DATASET: A SET OF EMAILS LABELED SPAM OR HAM

Our second dataset is a dataset of 100 emails. Out of them, 40 are spam, and the remaining 60 are ham. The labels in this dataset are "spam" and "ham," and the goal of a model would be to predict the label based on the features of the email.

A SUPER EFFECTIVE YET SUPER USELESS MODEL

Accuracy is a very useful metric, but does it paint the whole picture of the model? It doesn't, and we'll illustrate this with an example. For now, let's focus on the coronavirus dataset. We'll come back to the email dataset in the next section.

Suppose a data scientist tells us the following: "I have developed a test for coronavirus that takes 10 seconds to run, doesn't require any examinations, and has an accuracy of 99%!" Should we be excited or skeptical? We'd probably be skeptical. Why? We'll soon see that calculating a model's accuracy sometimes isn't enough. Our model may have an accuracy of 99% and yet be completely useless.

Can we think of a completely useless model that predicts coronavirus in our dataset, yet is correct 99% of the time? Recall that our dataset contains 1,000 patients, and out of those, 10 have coronavirus. Feel free to put this book down for a moment and think of how to build a model that detects coronavirus and that is correct 99% of the time for this dataset.

It could be a model like this: simply diagnose every patient as healthy. That's a simple model, but it's still a model; it's the model that predicts everything as one class.

What is the accuracy of this model? Well, out of 1,000 tries, it's incorrect 10 times and correct 990 times. This gives an accuracy of 99%, just like we promised. However, the model equates to telling everyone that they are healthy in the middle of a global pandemic, which is terrible!

What is the problem with our model, then? The problem is that errors are not created equal, and some mistakes are much more expensive than others, as we'll see in the next section.

How to fix the accuracy problem? Defining different types of errors and how to measure them

In the previous section, we built a useless model that had great accuracy. In this section, we study what went wrong. Namely, we study what the problem was with calculating accuracy in that model, and we introduce some slightly different metrics that will give us better evaluations of this model.

The first thing we need to study is types of errors. In the next section, we see that some errors are more critical than others. Then in the sections "Storing the correctly and incorrectly classified points in a table" to "Recall, precision, or F-scores," we learn different metrics that are more equipped to catch these critical errors than accuracy.

FALSE POSITIVES AND FALSE NEGATIVES: WHICH ONE IS WORSE?

In many cases, the total number of errors doesn't tell us everything about the model's performance, and we need to dig in deeper and identify certain types of errors in different ways. In this section, we see two types of errors. What are the two types of errors that the coronavirus model can make? It can diagnose a healthy person as sick or a sick person as healthy. In our model, we label the sick patients as positive, by convention. The two error types are called false positives and negatives, as follows:

- **False positive**: a healthy person who is incorrectly diagnosed as sick
- **False negative**: a sick person who is incorrectly diagnosed as healthy

In the general setting, a false positive is a data point that has a negative label, but the model falsely classifies it as positive. A false negative is a data point that has a positive label, but the model falsely classified it as negative. Naturally, the cases that are correctly diagnosed also have names, as follows:

- **True positive**: a sick person who is diagnosed as sick
- **True negative**: a healthy person who is diagnosed as healthy

In the general setting, a true positive is a data point that has a positive label that is correctly classified as positive, and a true negative is one with a negative label that is correctly classified as negative.

Now, let's look at the email dataset. Let's say we have a model that predicts whether each email is spam or ham. We consider the positives to be the spam emails. Therefore, our two types of errors follow:

- **False positive**: a ham email that is incorrectly classified as spam
- **False negative**: a spam email that is incorrectly classified as ham

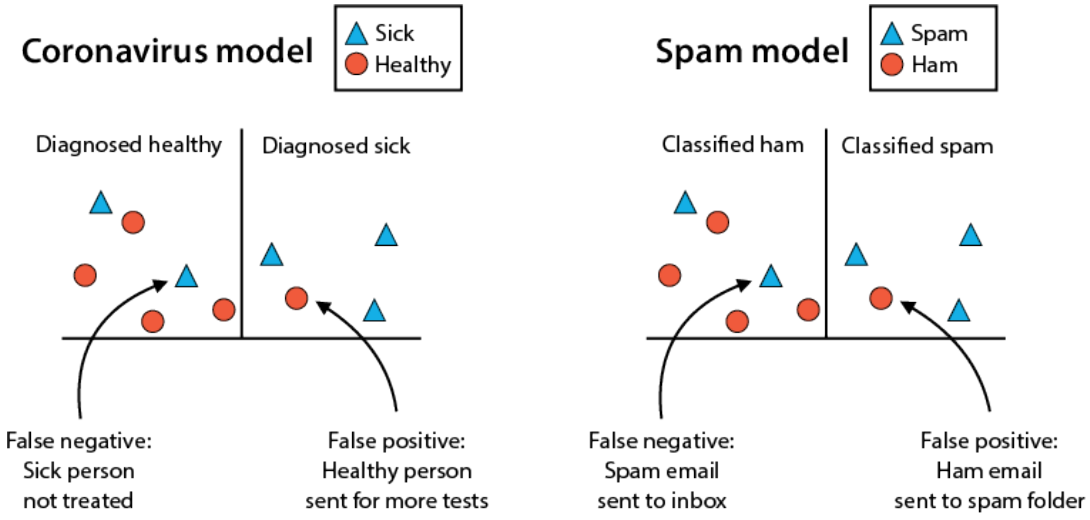
And the correctly classified emails are the following:

- **True positive**: a spam email that is correctly classified as spam
- **True negative**: a ham email that is correctly classified as ham

Figure 7.1 shows a graphical representation of the models, in which the vertical line is the boundary, the zone to the left of the line is the negative zone, and the zone to the right is the positive zone. The triangles are the points with positive labels, and the circles are the points with negative labels. The four quantities defined above are the following:

- Triangle to the right of the line: true positive
- Triangle to the left of the line: false negative
- Circle to the right of the line: false positive
- Circle to the left of the line: true negative

Figure 7.1 Examples of two models that are widely used in real life and that we'll use throughout this chapter. On the left, a coronavirus model where the people are diagnosed as healthy or sick. On the right, a spam-detection model where the emails are classified as spam or ham. For each model, we have highlighted some of their errors and separated them as false positives and false negatives.



Notice that both models in figure 7.1 produce the following quantities:

- Three true positives
- Four true negatives
- One false positive
- Two false negatives

To see the difference between the coronavirus model and the spam model, we need to analyze which one is worse between the false positives and the false negatives. Let's do this for each model separately.

ANALYZING FALSE POSITIVES AND NEGATIVES IN THE CORONAVIRUS MODEL

Let's stop and think. In the coronavirus model, which one sounds like a worse mistake: a false positive or a false negative? In other words, what is worse: to incorrectly diagnose a healthy patient as sick, or a sick patient as healthy? Let's say that when we diagnose a patient as healthy, we send them home with no treatment, and that when we diagnose a patient as sick, we send them for more tests. Incorrectly diagnosing a healthy person may be a small nuisance, because it means a healthy person will have to stay for extra tests. However, incorrectly diagnosing a sick person means that a sick person won't get treatment, their condition may worsen, and they may potentially infect many others. Thus, **in the coronavirus model, a false negative is much worse than a false positive**.

ANALYZING FALSE POSITIVES AND NEGATIVES IN THE SPAM EMAIL MODEL

Now we'll do the same analysis on the spam model. In this case, let's say that if our spam classifier classifies an email as spam, then this email is automatically deleted. If it classifies it as ham, then the email is sent to our inbox. Which one sounds like a worse mistake: a false positive or a false negative? In other words, what is worse, to incorrectly classify a ham email as spam and delete it, or to incorrectly classify a spam email as ham and send it to the inbox? I think we can agree that deleting a good email is much worse than sending a spam email to the inbox. The occasional spam email in our inbox can be annoying, but a deleted ham email can be a complete disaster! Imagine the sadness we would feel in our heart if our

grandma sent us a very kind email telling us she baked cookies, and our filter deleted it. Therefore, **in the spam email model, a false positive is much worse than a false negative**.

This is where the two models differ. In the coronavirus model, a false negative is worse, whereas in the spam email model, a false positive is worse. The problem with measuring the accuracy in any of these two models is that the accuracy considers both types of errors as equally serious and doesn't tell them apart.

In the section "A super effective yet super useless model," we had an example of a model that diagnosed every patient as healthy. This model made only 10 errors among 1,000 patients. However, all those 10 were false negatives, which is terrible. If those 10 were false positives instead, the model would be much better.

In the following sections, we'll devise two new metrics, similar to accuracy. The first metric helps us deal with models in which false negatives are worse, and the second one helps us deal with models in which false positives are worse.

STORING THE CORRECTLY AND INCORRECTLY CLASSIFIED POINTS IN A TABLE: THE CONFUSION MATRIX

In the previous subsection, we learned about false positives, false negatives, true positives, and true negatives. To keep track of these four entities, we put them together in a table aptly named *the confusion matrix.* For binary classification models (models that predict two classes), the confusion matrix has two rows and two columns. In the rows we write the true labels (in the medical example, this is the condition of the person, sick or healthy), and in the columns we write the predicted labels (the diagnosis of the person, sick or healthy). The general confusion matrix is illustrated in table 7.1, and specific ones for examples of models in these two datasets are shown in tables 7.2 to 7.5. This is called a confusion matrix because it makes it easy to see if the model is confusing two classes, namely the positive (sick) and the negative (healthy).

Table 7.1 The confusion matrix helps us count how many times each class is predicted correctly and how many times each class is confused with a different class. In this matrix, the rows represent the label, and the columns represent the prediction. The elements in the diagonal are classified correctly, and the elements off the diagonal are not. (view table figure)

Person's condition	Predicted positive	Predicted negative
Positive	Number of true positives	Number of false negatives
Negative	Number of false positives	Number of true negatives

For our existing model (the one that diagnoses every patient as healthy), which from now on we call coronavirus model 1, the confusion matrix is illustrated in table 7.2.

Table 7.2 The confusion matrix of our coronavirus model helps us dig into our model and tell the two types of errors apart. This model makes 10 false negative errors (a sick person diagnosed healthy) and zero false positive errors (a healthy person diagnosed sick). Notice that the model creates too many false negatives, which are the worst type of error in this case, which implies that this model is not very good. (view table figure)

Coronavirus model 1	Diagnosed sick (predicted positive)	Diagnosed healthy (predicted negative)
Sick (positive)	0 (number of true positives)	10 (number of false negatives)

Healthy (negative)	0 (number of false positives)	990 (number of true negatives)

For problems with more classes, we have a larger confusion matrix. For example, if our model classifies images into aardvarks, birds, cats, and dogs, then our confusion matrix is a four-by-four matrix, where along the rows we have the true labels (the type of animal), and along the columns we have the predicted labels (the type of animal that the model predicted). This confusion matrix also has the property that the correctly classified points are counted in the diagonal, and the incorrectly classified are counted off the diagonal.

RECALL: AMONG THE POSITIVE EXAMPLES, HOW MANY DID WE CORRECTLY CLASSIFY?

Now that we know the two types of errors, in this section, we learn a metric that will give coronavirus model 1 a much lower score. We have established that the problem with this model is that it gives us too many false negatives, namely, that it diagnoses too many sick people as healthy.

Let's assume, for a moment, that we don't mind false positives at all. Say that if the model diagnoses a healthy person as sick, the person may need to take an extra test or quarantine for a little longer, but this is no problem at all. Naturally, this is not the case; false positives are also expensive, but for now, let's pretend that they're not. In this case, we need a metric that replaces accuracy and that places importance on finding positive cases and cares less about mistakenly classifying negative cases.

To find this metric, we need to evaluate what our goal is. If we want to cure coronavirus, then what we really want is the following: out of all the sick people in the world, we want to find them all. It doesn't matter if we accidentally find others who aren't sick, as long as we find all the sick ones. This is the key. This new metric, called ***recall***, measures precisely that: out of the sick people, how many did our model diagnose correctly?

In more general lingo, recall finds the proportion of correct predictions among the data points with a positive label. This is the number of true positives, divided by the total number of positives. Coronavirus model 1 has a total of 0 true positives among 10 positives, so its recall is 0/10 = 0. Another way to put it is as the number of true positives divided by the sum of true positives and false negatives, as shown here:

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

In contrast, let's say we had a second model called coronavirus model 2. The confusion matrix of this model is shown in table 7.3. This second model made more mistakes than the first model—it made 50 total mistakes as opposed to only 10. The accuracy of the second model is 950/1000 = 0.95, or 95%. In terms of accuracy, the second model is not as good as the first model.

However, the second model correctly diagnosed eight out of the 10 sick people and 942 out of the 1,000 people. In other words, it has two false negatives and 48 false positives.

Table 7.3 The confusion matrix of our second coronavirus model (view table figure)

Coronavirus model 2	Diagnosed sick	Diagnosed healthy
Sick	8 (true positives)	2 (false negatives)
Healthy	48 (false positives)	942 (true negatives)

The recall of this model is the number of true positives (eight sick people correctly diagnosed) divided by the total number of positives (10 sick

people), which is 8/10 = 0.8, or 80%. In terms of recall, the second model is much better. Let's summarize these calculations for clarity as follows:

Coronavirus Model 1:

True positives (sick patients diagnosed sick and sent for more tests) = 0

False negatives (sick patients diagnosed healthy and sent home) = 10

Recall = 0/10 = 0%

Coronavirus Model 2:

True positives (sick patients diagnosed sick and sent for more tests) = 8

False negatives (sick patients diagnosed healthy and sent home) = 2

Recall = 8/10 = 80%

Models like the coronavirus model, in which false negatives are much more expensive than false positives, are ***high recall models***.

Now that we have a better metric, could we fool this metric in the same way we fooled accuracy? In other words, can we build a model that has total recall? Well, get ready for a surprise, because we can. If we build a model that diagnoses every patient as sick, this model has a 100% recall. However, this model is terrible, too, because although it has zero false negatives, it has too many false positives to make it a good model. It seems that we still need more metrics to be able to evaluate our models properly.

PRECISION: AMONG THE EXAMPLES WE CLASSIFIED AS POSITIVE, HOW MANY DID WE CORRECTLY CLASSIFY?

In the previous section we learned recall, a metric that measures how well our model did with false negatives. That metric worked well for the coronavirus model—we've seen that this model can't afford to have too many false negatives. In this section, we learn about a similar metric,

precision, which measures how well our model does with false positives. We'll use this metric to evaluate the spam email model, because this model can't afford to have too many false positives.

Just as we did with recall, to come up with a metric, we first need to define our goal. We want a spam filter that doesn't delete any ham emails. If instead of deleting emails, it sends them to a spam box. Then we need to look into that spam box and hope that we do not see a single ham email. Thus, our metric should measure precisely that: Among the emails in our spam box, how many were actually spam? In other words, out of the emails that are predicted to be spam, how many of them are actually spam? This is our metric, and we call it *precision*.

More formally, precision considers only the data points that have been labeled positive, and among those, how many are true positives. Because the data points that are predicted positive are the union of the true positives and the false positives, the formula is the following:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

Remember that in our dataset of 100 emails, 40 are spam and 60 are ham. Say we trained the following two models called spam model 1 and spam model 2. Their confusion matrices are shown in tables 7.4 and 7.5.

Table 7.4 The confusion matrix of our first spam model (view table figure)

Spam model 1	Predicted spam	Predicted ham
Spam	30 (true positives)	10 (false negatives)
Ham	5 (false positives)	55 (true negatives)

Table 7.5 The confusion matrix of our second spam model (view table figure)

Spam model 2	Predicted spam	Predicted ham
Spam	35 (true positives)	5 (false negatives)
Ham	10 (false positives)	50 (true negatives)

In terms of accuracy, it seems that both models are just as good—they both make correct predictions 85% of the time (85 correct out of 100 emails). However, at first glance, it seems that the first model is better than the second one, because the first model deletes only five ham emails, and the second one deletes 10 of them. Now let's calculate the precision as follows:

Spam Model 1:

- True positives (spam emails deleted) = 30
- False positives (ham emails deleted) = 5
- Precision = 30/35 = 85.7%

Spam Model 2:

- True positives (spam emails deleted) = 35
- False positives (ham emails deleted) = 10
- Precision = 35/45 = 77.7%

Just as we thought: precision gave a higher score to the first model than to the second model. We conclude that models like the spam model, in which false positives are much more expensive than false negatives, are ***high precision models***. And why is the first model better than the second one? The second model deleted 10 good (ham) emails, but the first model deleted only five of them. The second model may have cleaned up more

spam than the first one, but that doesn't make up for the five ham emails it deleted.

Now, in the same way we tricked accuracy and recall, we can also trick precision. Consider the following spam filter: a spam filter that never detects any spam. What is the precision of this model? This is complicated, because there are zero spam emails deleted (zero true positives) and zero ham emails deleted (zero false positives). We won't attempt to divide zero over zero, because this book would burst into flames, but by convention, a model that makes no false positive mistakes has a precision of 100%. But, of course, a spam filter that does nothing is not a good spam filter.

This goes to show that no matter how good our metrics are, they can always be fooled. That doesn't mean they don't work. Accuracy, precision, and recall are useful tools in a data scientist's toolbox. It is up to us to decide which ones are good for our model, by deciding what errors are more expensive than others. Always be careful to not fall into the trap of thinking that a model is good before evaluating it with different metrics.

COMBINING RECALL AND PRECISION AS A WAY TO OPTIMIZE BOTH: THE F-SCORE

In this section, we discuss the F-score, a metric that combines both recall and precision. In the previous sections, we saw two examples, the coronavirus model and the spam model, in which either false negatives or false positives were more important. However, in real life, both are important, even if they are important to different degrees. For example, we may want a model that doesn't misdiagnose any sick person but that also doesn't misdiagnose too many healthy people, because misdiagnosing a healthy person may involve unnecessary and painful testing, or even an unnecessary surgery, which could affect their health negatively. In the same way, we may want a model that doesn't delete any of our good emails. But to be a good spam filter, it still needs to catch a lot of spam; otherwise, it's useless. The F-score has a parameter β accompanying it, so the more common term is F_β-score. When $\beta = 1$, it is called the F_1-score.

CALCULATING THE F-SCORE

Our goal is to find a metric that gives us some number between the recall and the precision. The first thing that comes to mind is the average between recall and precision. Would this work? It would, but it's not the one we pick, for one fundamental reason. A good model is one that has good recall and good precision. If a model has, say, recall of 50% and precision of 100%, the average is 75%. This is a good score, but the model may not be, because a recall of 50% is not very good. We need a metric that behaves like the average but that is closer to the minimum value of the two.

A quantity that is like the average of two numbers is called the *harmonic mean*. Whereas the average of two numbers a and b is $(a + b)/2$, their harmonic mean is $2ab/(a + b)$. The harmonic mean has this property: it is always smaller than or equal to the average. If the numbers a and b are equal, one can quickly check that their harmonic mean is equal to both of them, just like the average. But in other cases, the harmonic mean is smaller. Let's look at an example: If $a = 1$ and $b = 9$, their average is 5. The harmonic mean is $\frac{2 \cdot 1 \cdot 9}{1+9} = 1.8$

The F_1-score is defined as the harmonic mean between the precision and the recall, as follows:

$$F_1 = \frac{2PR}{P+R}$$

If both numbers are high, the F_1-score is high. However, if one of them is low, the F_1-score will be low. The purpose of the F_1-score is to measure if both recall and precision are high and to ring a bell when one of these two scores is low.

CALCULATING THE F_B-SCORE

In the previous subsection, we learned about the F_1-score, a score that combines recall and precision, for the purpose of evaluating a model.

However, sometimes we want more recall than precision, or vice versa. Thus, when we combine the two scores, we may want to give one of them more weight. This means that sometimes we may want a model that cares both about false positives and false negatives but assigns more weight to one of them. For example, the coronavirus model cares much more about false negatives, because people's lives may depend on a correct identification of the virus, but it still doesn't want to create too many false positives, because we may not want to spend excessive resources retesting healthy people. The spam model cares much more about false positives, because we really wouldn't want to delete good emails but still doesn't want to create too many false negatives, because we wouldn't want our inbox cluttered with spam messages.

This is where F_β-score comes into play. The formula for the F_β-score may look complicated at first, but once we look at it carefully, it does exactly what we want. The F_β-score uses a parameter called β (the Greek letter beta), which can take any positive value. The point of β is to act as a dial that we turn to emphasize precision or recall. More specifically, if we slide the β dial to zero, we get full precision; if we slide it to infinity, we get full recall. In general, the lower the value of β, the more we emphasize precision, and the higher the value of β, the more we emphasize recall.

This is where F_β-score is defined as follows (where precision is P and recall is R):

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R}$$

Let's analyze this formula carefully by looking at some values for β.

Case 1 $\beta = 1$

When β is equal to 1, the F_β-score becomes the following:

$$F_1 = \frac{(1+1^2)PR}{1^2P+R}$$

This is the same as the F_1-score that considers recall and precision equally.

Case 2 β = 10

When β is equal to 10, the F_β-score becomes the following:

$$F_{10} = \frac{(1+10^2)PR}{10^2P+R}$$

This can be written as

$$\frac{101PR}{100P+R}.$$

This is similar to the F_1-score, except notice how it gives much more importance to R than to P. To see this, notice that the limit as β tends to ∞ of the F_β-score is R. Therefore, when we want a score between recall and precision that gives more weight to recall, we pick a value of β that is larger than 1. The larger the value, the more emphasis we put on the recall and less on the precision.

Case 3 β = 0.1

When β is equal to 0.1, the F_β-score becomes the following:

$$F_{0.1} = \frac{\left(1+0.1^2\right)PR}{0.1^2P+R}$$

Just like before, we can write this as

$$\frac{1.01PR}{0.01P+R}.$$

This is similar to the formula from case 2, except this one gives P a lot more importance. Therefore, when we want a score between recall and precision that gives more weight to precision, we pick a value of β that is smaller than 1. The smaller the value, the more emphasis we put on the precision and less on the recall. In the limits, we say that a value of $\beta = 0$ gives us the precision, and a value of $\beta = \infty$ gives us the recall.

RECALL, PRECISION, OR F-SCORES: WHICH ONE SHOULD WE USE?

Now, how do we put recall and precision into practice? When we have a model, is it a high recall or a high precision model? Do we use the F-score? If so, which value of βshould we pick? The answers to these questions are up to us, the data scientists. It is important for us to know the problem we are trying to solve very well to decide which error, between a false positive and a false negative, is more expensive.

In the previous two examples, we can see that because the coronavirus model needs to focus more on recall than on precision, we should pick a large value of *b*, say, for example, 2. In contrast, the spam model needs to focus more on precision than on recall, so we should pick a small value of β, say, 0.5. For more practice analyzing models and estimating what values of β to use, see exercise 7.4 at the end of the chapter.

A useful tool to evaluate our model: The receiver operating characteristic (ROC) curve

In the section "How to fix the accuracy problem?," we learned how to evaluate a model using metrics such as precision, recall, and the F-score. We also learned that one of the main challenges of evaluating a model lies in the fact that more than one type of error exists and different types of errors have different levels of importance. We learned two types of errors: false positives and false negatives. In some models, false negatives are much more expensive than false positives, and in some models, it's the opposite.

In this section, I teach you a useful technique to evaluate a model based on its performance on false positives and negatives at the same time. Furthermore, this method has an important feature: a dial that allows us to gradually switch between a model that performs well on false positives and one that performs well on false negatives. This technique is based on a curve called the *receiver operating characteristic (ROC) curve*.

Before we learn the ROC curve, we need to introduce two new metrics called specificity and sensitivity. Actually, only one of them is new. The other one, we've seen before.

SENSITIVITY AND SPECIFICITY: TWO NEW WAYS TO EVALUATE OUR MODEL

In the section "How to fix the accuracy problem?," we defined recall and precision as our metrics and found that they were useful tools to measure our model both for false negatives and for false positives. However, in this section, we use two different, yet very similar, metrics: ***sensitivity*** and ***specificity***. These have a similar use to the previous ones, but they are more useful for us when we have to build the ROC curve. Furthermore, although precision and recall are more widely used by data scientists, sensitivity and specificity are more common in the medical field. Sensitivity and specificity are defined as follows:

Sensitivity (true positive rate): the capacity of the test to identify the positively labeled points. This is the ratio between the number of true positives and the total number of positives. (Note: this is the same as recall).

$$Sensitivity = \frac{True\ positives}{True\ positives + False\ negatives}$$

Specificity (true negative rate): the capacity of the test to identify the negatively labeled points. This is the ratio between the number of true negatives and the total number of negatives.

$$Specificity = \frac{True\ \ negatives}{True\ \ negatives + \ \ False\ \ positives}$$

As I mentioned, sensitivity is the same as recall. However, specificity is not the same as precision (each nomenclature is popular in different disciplines, and for that reason, we use them both here). We see this more in detail in the section "Recall is sensitivity, but precision and specificity are different."

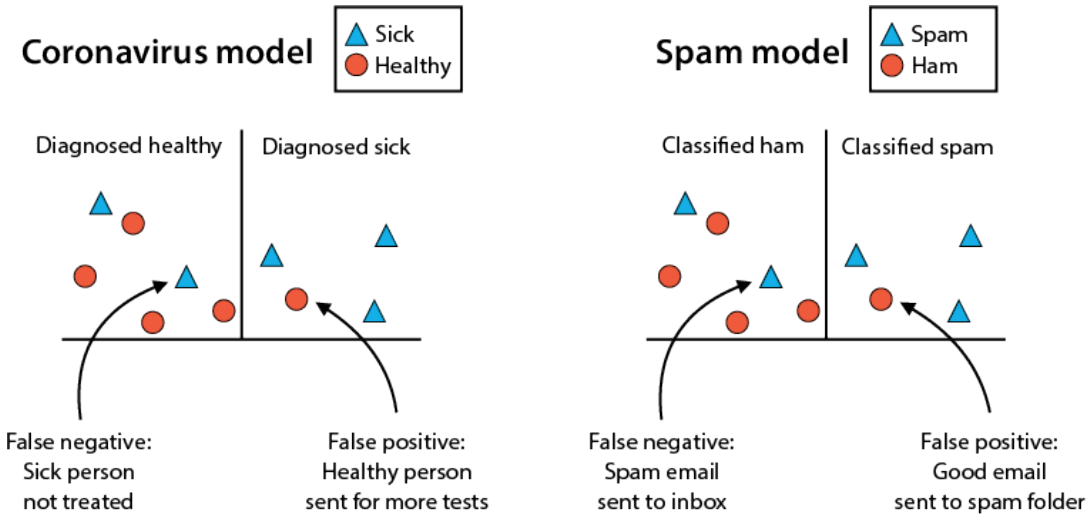
In the coronavirus model, the sensitivity is the proportion of sick people that the model has correctly diagnosed, among all the sick people. The specificity is the proportion of healthy people the model has correctly diagnosed, among the healthy people. We are more concerned about correctly diagnosing sick people, so we need the coronavirus model to have *high sensitivity*.

In the spam-detection model, the sensitivity is the proportion of spam messages we correctly deleted, among all the spam messages. The specificity is the proportion of ham emails we correctly sent to the inbox,

among all the ham emails. Because we are more concerned about correctly detecting the ham emails, we need the spam detection model to have *high specificity*.

To clarify the previous concepts, let's look at them in the graphical example we are working on. Namely, let's calculate the specificity and sensitivity for our two models in figure 7.2 (which is the same as figure 7.1).

Figure 7.2 On the left, a coronavirus model where the people are diagnosed as healthy or sick; on the right, a spam detection model where the emails are classified as spam or ham



As we saw previously, these two models produce the following quantities:

- Three true positives
- Four true negatives
- One false positive
- Two false negatives

Now let's calculate the specificity and sensitivity of these models.

CALCULATING THE SENSITIVITY

In this case, we calculate sensitivity as follows: among the positive points, how many did the model classify correctly? This is equivalent to asking: among the triangles, how many are located to the right of the line? There

are five triangles, and the model classified three of them correctly to the right of the line, so the sensitivity is 3/5, which equals 0.6, or 60%.

CALCULATING THE SPECIFICITY

We calculate specificity as follows: among the negative points, how many did the model classify correctly? This is equivalent to asking: among the circles, how many are located to the left of the line? There are five circles, and the model classified four of them correctly to the left of the line, so the specificity is 4/5, which equals 0.8, or 80%.

THE RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE: A WAY TO OPTIMIZE SENSITIVITY AND SPECIFICITY IN A MODEL

In this section, we see how to draw the receiver operating characteristic (ROC) curve, which will give us a lot of information about the model. In short, what we'll do is slowly modify the model and record the sensitivity and specificity of the model at each time step.

The first and only assumption we need to make about our model is that it returns the prediction as a continuous value, namely, as a probability. This is true about models such as logistic classifiers, where the prediction is not a class, such as positive/negative, but a value between 0 and 1, such as 0.7. What we normally do with this value is pick a threshold, such as 0.5, and classify every point that receives a prediction higher than or equal to the threshold as positive and every other point as negative. However, this threshold can be any value—it need not be 0.5. Our procedure consists in varying this threshold from 0 all the way to 1 and recording the sensitivity and specificity of the model at each threshold value.

Let's look at an example. We calculate the sensitivity and specificity for three different thresholds: 0.2, 0.5, and 0.8. In figure 7.3, we can see how many points are to the left and right of the line for each one of these thresholds. Let's study them in detail. Remember that sensitivity is the ratio of true positives over all positives, and specificity is the ratio of true negatives over all negatives. Also remember that for each one of these, there are five total positives and five total negatives.

Threshold = 0.2

Number of true positives: 4

Sensitivity: 4/5

Number of true negatives: 3

Specificity: 3/5

Threshold = 0.5

Number of true positives: 3

Sensitivity: 3/5

Number of true negatives: 4

Specificity: 4/5

Threshold = 0.2

Number of true positives: 2

Sensitivity: 2/5

Number of true negatives: 5

Specificity: $^{5}/_{5} = 1$

Note that a low threshold leads to many positive predictions. Therefore, we will have few false negatives, implying a high sensitivity score, and many false positives, implying a low specificity score. Similarly, a high threshold implies a low sensitivity score and a high specificity score. As we move the threshold from low to high, the sensitivity decreases, and the specificity increases. This is an important point that we'll touch on later in this

chapter, when we get to the point of deciding the best threshold for our model.

Now we are ready to build the ROC curve. First, we consider a threshold of 0 and slowly increase the value of this threshold by small intervals, until it reaches 1. For every increment in threshold, we pass over exactly one point. The values of the thresholds are not important—what is important is that at every step, we pass over exactly one point (this is possible because all the points give us different scores, but it's not a requirement in general). Thus, we'll refer to the steps as 0, 1, 2,..., 10. In your head, you should imagine the vertical line in figure 7.3 starting at 0 and moving slowly from left to right, sweeping one point at a time, until reaching 1. These steps are recorded in table 7.6, together with the number of true positives and negatives, sensitivity, and specificity at each step.

One thing to notice is that in the first step (step 0), the line is at threshold 0. This means every point is classified as positive by the model. All the positive points are also classified as positive, so every positive is a true positive. This means that at timestep 0, the sensitivity is 5/5 = 1. But because every negative point is classified as positive, there are no true negatives, so the specificity is 0/5 = 0. Similarly, at the last step (step 10), the threshold is 1, and we can check that because every point is classified as negative, the sensitivity is now 0 and the specificity is 1. For clarity, the three models in figure 7.3 are highlighted in table 7.6 as timesteps 4, 6, and 8, respectively.

Figure 7.3 The effects of moving the threshold on the sensitivity and the specificity. On the left, we have a model with a low threshold; in the middle, we have one with a medium threshold; and on the right, we have one with a high threshold. For each of the models, there are five positive and five negative points. Each model is represented by the vertical line. The model predicts that the points to the right of the line are positive and those to the left are negative. For each of the models, we've counted the number of true positives and true negatives, that is, the number of positive and negative points that have been correctly predicted. We have used those to calculate the sensitivity and the specificity. Note that as we increase the threshold (i.e., as we move the vertical line from left to right), the sensitivity goes down and the specificity goes up.

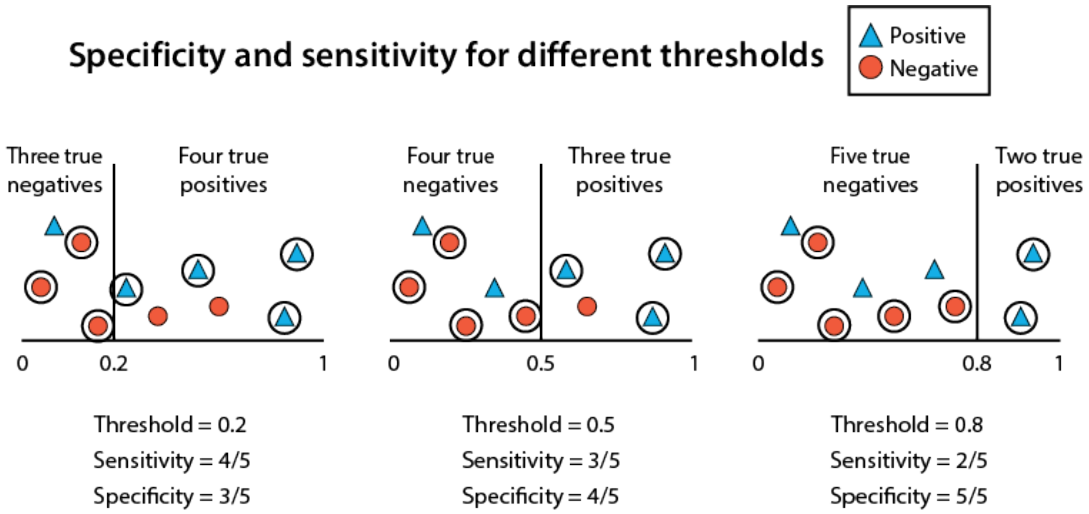


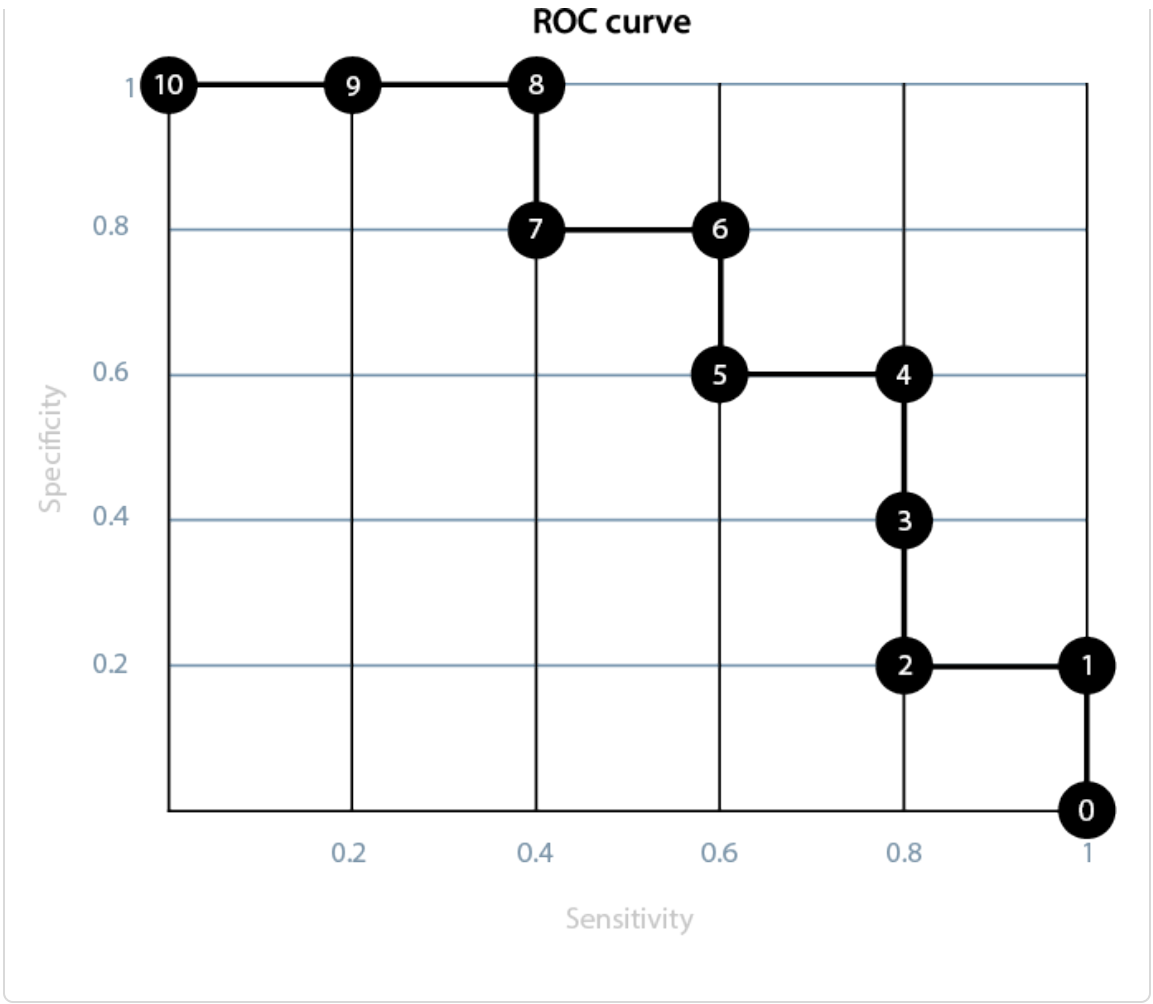
Table 7.6 All the timesteps in the process of increasing our threshold, which is an important step in building our ROC curve. At each timestep, we record the number of true positives and true negatives. We then calculate the specificity of the model by dividing the number of true positives by the total number of positives. As a final step, we calculate the specificity by dividing the number of true negatives by the total number of negatives. **(view table figure)**

Step	True positives	Sensitivity	True negatives	Specificity
0	5	1	0	0
1	5	1	1	0.2
2	4	0.8	1	0.2
3	4	0.8	2	0.4
4	4	0.8	3	0.6
5	3	0.6	3	0.6
6	3	0.6	4	0.8
7	2	0.4	4	0.8

8	2	0.4	5	1
9	1	0.2	5	1
10	0	0	5	1

As a last step, we plot the sensitivity and specificity values. This is the ROC curve, which we see in figure 7.4. In this figure, each of the black points corresponds to a timestep (indicated inside the point), the horizontal coordinate corresponds to the sensitivity, and the vertical coordinate to the specificity.

Figure 7.4 Here we can see the ROC curve corresponding to our ongoing example, which gives us a great deal of information on our model. The highlighted dots correspond to the timesteps obtained by moving our threshold from 0 to 1, and each dot is labeled by the timestep. On the horizontal axis we record the sensitivity of the model at each timestep, and on the vertical axis we record the specificity.



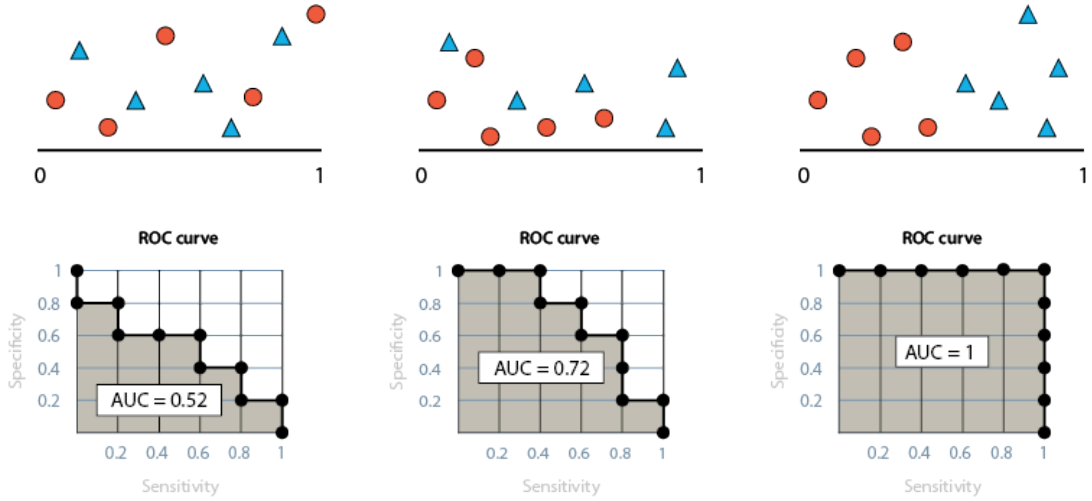
A METRIC THAT TELLS US HOW GOOD OUR MODEL IS: THE AUC (AREA UNDER THE CURVE)

As we've seen before in this book, evaluating a machine learning model is a highly important task, and in this section, we discuss how to use the ROC curve to evaluate a model. For this, we've done all the work already—all that is left is to calculate the area under the curve, or AUC. At the top of figure 7.5, we can see three models, in which the prediction is given by the horizontal axis (from 0 to 1). On the bottom, you can see the three corresponding ROC curves. Each one of the squares has size 0.2 times 0.2. The number of squares under each curve are 13, 18, and 25, which amounts to areas under the curve of 0.52, 0.72, and 1.

Note that the best a model can do is an AUC of 1, which corresponds to the model on the right. The worst a model can do is an AUC of 0.5, because this means the model is as good as random guessing. This corresponds to the

model on the left. The model in the middle is our original model, with an AUC of 0.72.

Figure 7.5 In this figure, we can see that AUC, or area under the curve, is a good metric to determine how good a model is. The higher the AUC, the better the model. On the left, we have a bad model with an AUC of 0.52. In the middle, we have a good model with an AUC of 0.72. On the right, we have a great model with an AUC of 1.



What about a model with an AUC of zero? Well, this is tricky. A model with an AUC of zero would correspond to a model that classifies every point wrong. Is this a bad model? It's actually a very good model, because all we have to do to fix it is to flip all the positive and negative predictions and get a perfect model. It's the same effect as having a person that lies every single time they get a true-or-false question. All we have to do to get them to tell the truth is to flip all their answers. This means the worst we can have in a binary classification model is an AUC of 0.5, because this corresponds to a person who lies 50% of the time. They give us no information because we never know if they are telling the truth or lying! Incidentally, if we have a model with an AUC less than 0.5, we can flip the positive and negative predictions and obtain a model with an AUC larger than 0.5.

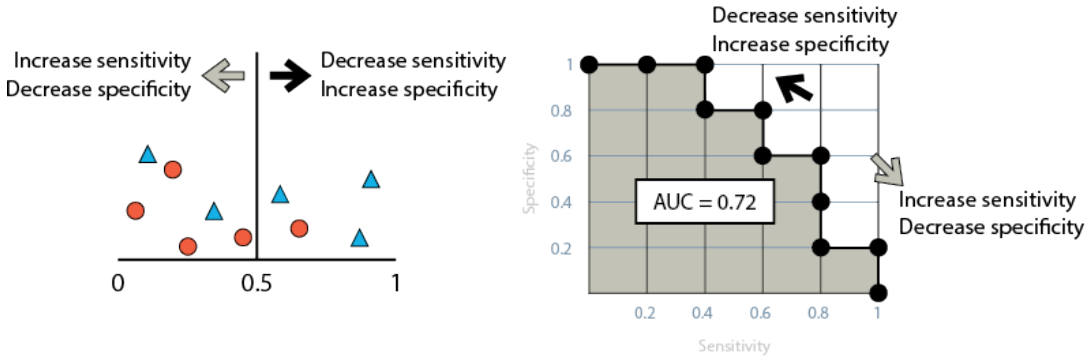
HOW TO MAKE DECISIONS USING THE ROC CURVE

The ROC is a powerful graphic that gives us a lot of information about our model. In this section, we learn how we can use it to improve our model. In short, we use the ROC to tweak the threshold in a model and apply it to pick the best model for our use case.

At the beginning of this chapter, we introduced two models, the coronavirus model and the spam-detection model. These models were very different because, as we saw, the coronavirus model requires high sensitivity, whereas the spam-detection model requires high specificity. Every model requires some amount of sensitivity and specificity based on the problem we are to solve. Let's say we are in the following situation: we are training a model that is supposed to have high sensitivity, and we get a model with low sensitivity and high specificity. Is there any way we can trade off some specificity and gain some sensitivity?

The answer is yes! We can trade off specificity and sensitivity by moving the threshold. Recall that when we first defined the ROC curve, we noticed that the lower the threshold, the higher sensitivity and lower specificity in the model, and the higher the threshold, the lower sensitivity and higher specificity in the model. When the vertical line corresponding to the threshold is at the very left, every point is predicted to be positive, so all the positives are true positives, whereas when the vertical line is at the very right, every point is predicted to be negative, so all the negatives are true negatives. As we move the line to the right, we lose some true positives and gain some true negatives, thus the sensitivity decreases and the specificity increases. Notice that as the threshold moves from 0 to 1, we move up and to the left in the ROC curve, as figure 7.6 illustrates.

Figure 7.6 The threshold of the model has a lot to do with the sensitivity and the specificity, and this relationship will help us pick the perfect threshold for our model. On the left, we have our model and, on the right, the corresponding ROC curve. As we increase or decrease the threshold, we change the sensitivity and specificity of the model, and this change is illustrated by moving in the ROC curve.



Why does this happen? The threshold tells us where we draw the line on classifying a point. For example, in the coronavirus model, the threshold tells us where we draw the line on a person being sent for more tests or sent home. A model with a low threshold is a model that sends people for extra tests if they so much as show mild symptoms. A model with a high threshold is one that needs the people to show strong symptoms to send them for more tests. Because we want to catch all the sick people, we want a low threshold for this model, which means we want a model with high sensitivity. For clarity, in figure 7.7, we can see the three thresholds used previously, as well as the points where they correspond in the curve.

If we want our model to have high sensitivity, we just push the threshold to the left (i.e., decrease it) until we get to a point in the curve that has as much sensitivity as we want. Note that the model may lose some specificity, and that's the price we pay. In contrast, if we want higher specificity, we push the threshold to the right (i.e., increase it) until we get to a point in the curve that has as much specificity as we want. Again, we lose some sensitivity during this process. The curve tells us exactly how much of one we gain and lose, so as data scientists, this is a great tool to help us decide the best threshold for our model. In figure 7.8, we can see a more general example with a bigger dataset.

Figure 7.7 The parallel between the threshold of the model and its ROC. The model on the left has a high threshold, low sensitivity, and high specificity. The model in the middle has medium values for threshold, sensitivity, and specificity. The model on the right has a low threshold, high sensitivity, and low specificity.

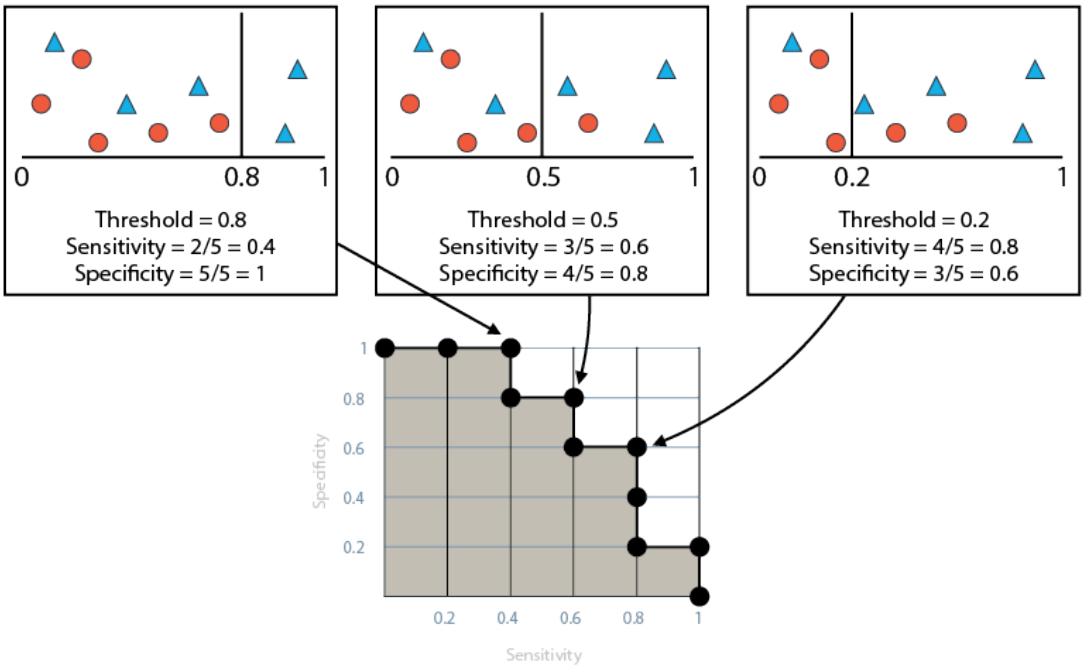
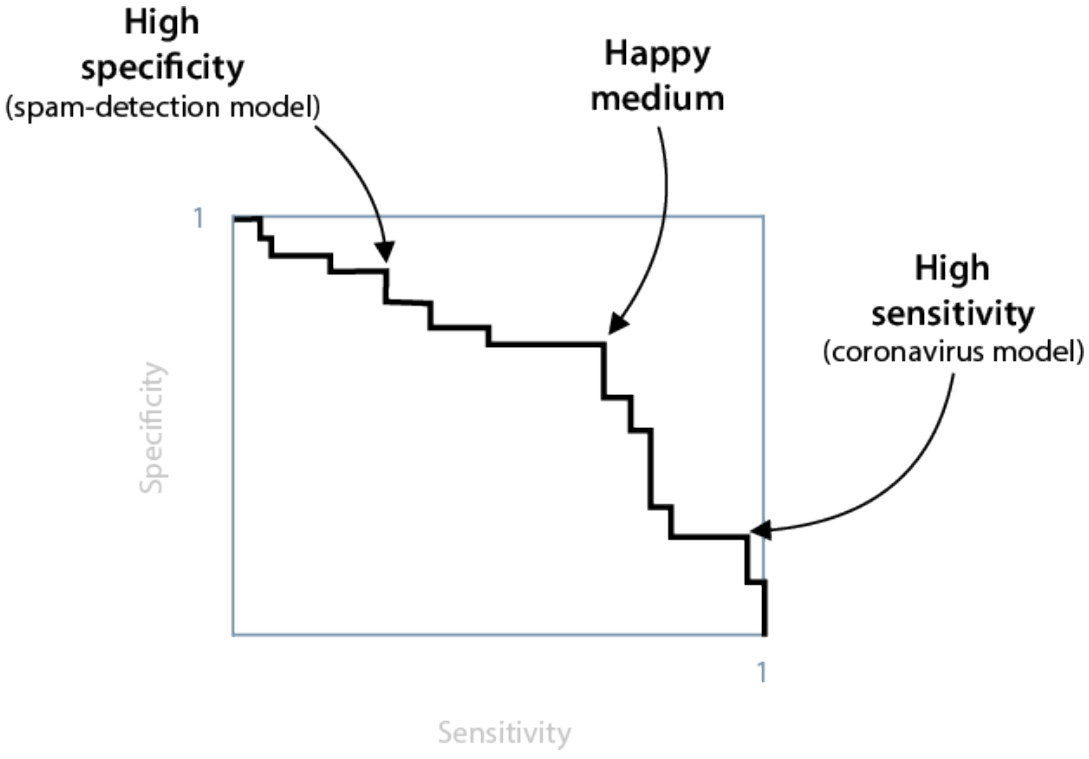


Figure 7.8 In this more general scenario, we can see an ROC curve and three points on it corresponding to three different thresholds. If we want to pick a threshold that gives us high specificity, we pick the one on the left. For a model with high sensitivity, we pick the one on the right. If we want a model that has a good amount of both sensitivity and specificity, we pick the one in the middle.



If we need a high sensitivity model, such as the coronavirus model, we would pick the point on the right. If we need a high specificity model, such as the spam-detection model, we may pick the point on the left. However, if we want relatively high sensitivity and specificity, we may go for the point in the middle. It's our responsibility as data scientists to know the problem well enough to make this decision properly.

RECALL IS SENSITIVITY, BUT PRECISION AND SPECIFICITY ARE DIFFERENT

At this point you may be wondering how we can remember all these terms off the top of our head. The answer is, they're hard not to get confused. Most data scientists (including the author) often need to quickly look them up in Wikipedia to make sure they're not confusing them. We could use a mnemonic to help us remember which one is which.

For example, when we think of recall, think of a car company that made a car with a fatal design flaw. They need to find all the faulty cars and ***recall*** them. If they accidentally get more cars that are not faulty, they simply return them. However, not finding one of the faulty cars would be terrible. Thus, recall cares about finding all the positively labeled examples. This represents a model with high ***recall***.

On the other hand, if we work for this car company, and we went a little overboard and started recalling ***all*** the cars, our boss may come over and say, "Hey, you are sending too many cars to fix, and we are running out of resources. Can you please be more selective and send me ***precisely*** those that are faulty?" Then we need to add precision to the model and try to find only the ones that are faulty, even if we accidentally miss some of the faulty ones (hopefully not!). This represents a model with high ***precision***.

When it comes to specificity and sensitivity, think of an earthquake sensor that beeps every time there is an earthquake. This sensor is tremendously ***sensitive***. If a butterfly sneezes in the next house, the sensor beeps. This sensor will capture all the earthquakes for sure, but it will also capture many other things that are not an earthquake. This represents a model with high ***sensitivity***.

Now, let's imagine that this sensor has a dial, and we turn its sensitivity all the way down. Now the sensor will beep only when there's a lot of movement. When that sensor beeps, we ***know*** it's an earthquake. The problem is that it may miss some smaller or medium earthquakes. In other words, this sensor is very ***specific*** to earthquakes, so it will most likely not beep with anything else. This represents a model with high ***specificity***.

If we go back and read the previous four paragraphs, we may notice the following two things:

- Recall and sensitivity are very similar.
- Precision and specificity are very similar.

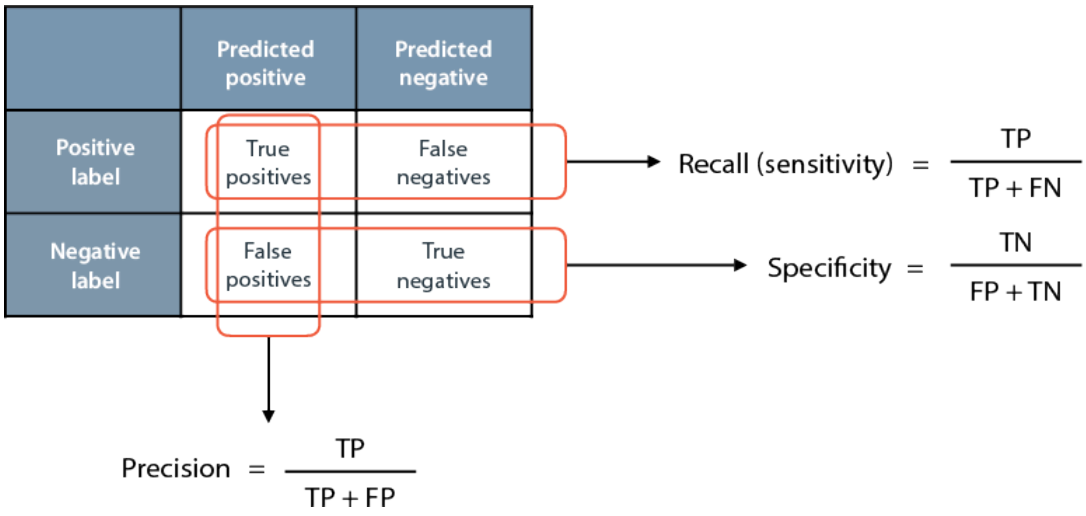
At the very least, recall and sensitivity have the same purpose, which is measuring how many false negatives there are. Similarly, precision and specificity also have the same purpose, which is measuring how many false positives there are.

It turns out that recall and sensitivity are ***exactly*** the same thing. However, precision and specificity are not the same thing. Although they don't measure the same, they both punish models with a high number of false positives. How to remember all these metrics? A graphical heuristic can help us remember recall, precision, sensitivity, and specificity. In figure 7.9, we see a confusion matrix with the four quantities: true positives, true negatives, false positives, and false negatives. If we focus on the top row (the positively labeled examples), we can calculate recall by dividing the number in the left column by the sum of the numbers in both columns. If we focus on the leftmost column (the examples that are predicted as positive), we can calculate precision by dividing the number on the top row by the sum of the numbers in both rows. If we focus on the bottom row (the negatively labeled examples), we can calculate specificity by dividing the number on the right column by the sum of the numbers on both columns. In other words

- Recall and sensitivity correspond to the top row.
- Precision corresponds to the left column.

- Specificity corresponds to the bottom row.

Figure 7.9 The top row of the confusion matrix gives us recall and sensitivity: the ratio between the number of true positives and the sum of true positives and false negatives. The leftmost column gives us precision: the ratio between the number of true positives and the sum of true positives and false positives. The bottom row gives us specificity: the ratio between the number of false positives and the sum of false positives and true negatives.



To wrap up, these quantities are the following in both of our models:

Medical model:

- **Recall and sensitivity**: among the sick people (positives), how many were correctly diagnosed as sick?
- **Precision**: among the people diagnosed as sick, how many were actually sick?
- **Specificity**: among the healthy people (negatives), how many were correctly diagnosed as healthy?

Email model:

- **Recall and sensitivity**: among the spam emails (positives), how many were correctly deleted?

- **Precision**: among the deleted emails, how many were actually spam?
- **Specificity**: among the ham emails (negatives), how many were correctly sent to the inbox?

Summary

- Being able to evaluate a model is as important as being able to train one.
- We can use several important metrics to evaluate a model. The ones we learned in this chapter are accuracy, recall, precision, F-score, specificity, and sensitivity.
- Accuracy calculates the ratio between correct predictions and total predictions. It is useful but can fail in certain cases, especially when the positive and negative labels are unbalanced.
- Errors are divided into two categories: false positives and false negatives.
    - A false positive is a negatively labeled point, which the model incorrectly predicts as positive.
    - A false negative is a positively labeled point, which the model incorrectly predicts as negative.
- For some models, false negatives and false positives are given different levels of importance.
- Recall and precision are useful metrics to evaluate models, especially when the models give false negatives and false positives different levels of importance.
    - Recall measures how many of the positive points were correctly predicted by the model. Recall is low when the model creates many false negatives. For this reason, recall is a useful metric in models in which we don't want many false negatives, such as models for medical diagnosis.
    - Precision measures how many of the points that the model predicted as positive are actually positive. Precision is low when the model creates many false positives. For this reason, precision is a useful metric in models in which we don't want many false positives, such as spam email models.
- F_1-score is a useful metric that combines recall and precision. It returns a value in between recall and precision but which is

closer to the smaller of the two.

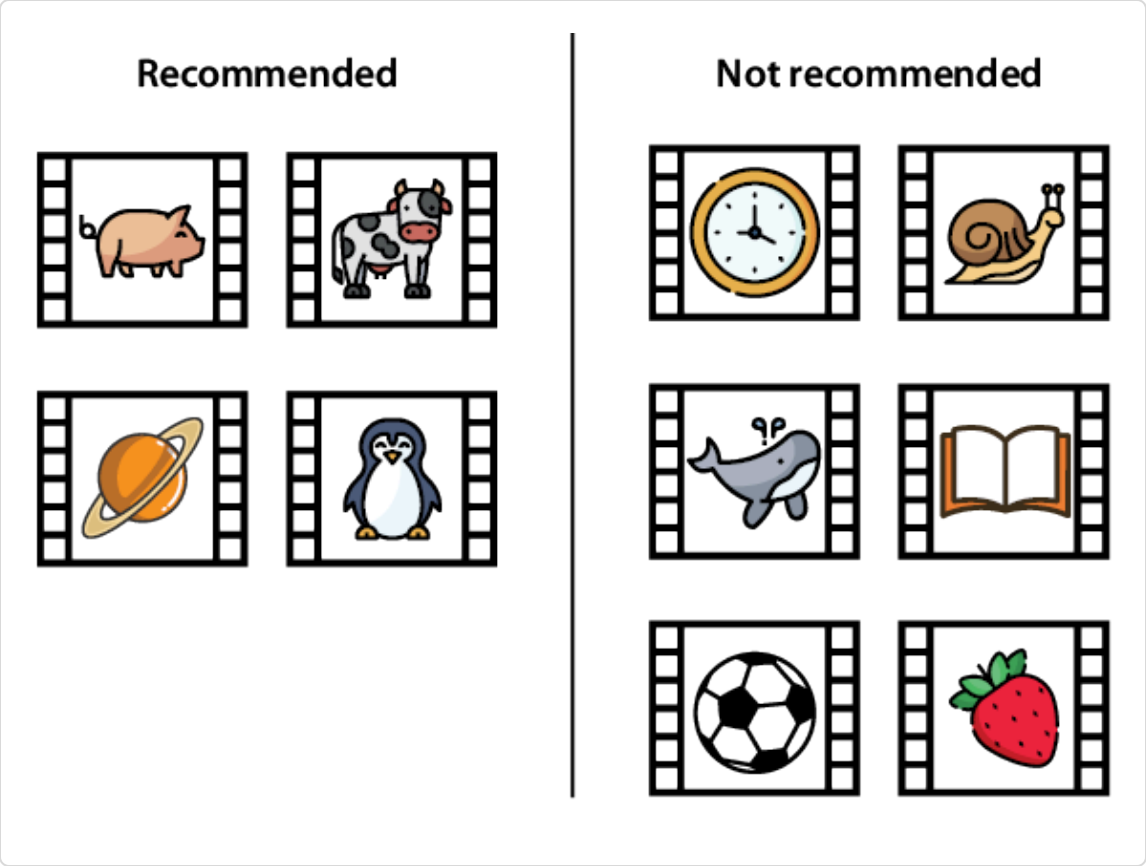
- F_β-score is a variation of F_1-score, in which one can adjust the parameter β to give either precision or recall a higher importance. Higher values of β give recall more importance, and lower values of β give precision more importance. F_β-score is particularly useful to evaluate models in which either precision or recall is more important than the other one, but we still care about both metrics.
- Sensitivity and specificity are two useful metrics that help us evaluate models. They are highly used in medical fields.
    - Sensitivity, or true positive ratio, measures how many of the positive points were correctly predicted by the model. Sensitivity is low when the model creates many false negatives. For this reason, sensitivity is a useful metric to use in medical models where we don't want to accidentally leave many healthy patients without treatment.
    - Specificity, or true negative ratio, measures how many of the negative points were correctly predicted by the model. Specificity is low when the model creates many false positives. For this reason, specificity is a useful metric in medical models where we don't want to accidentally treat or do further invasive tests on patients who are healthy.
- Recall and sensitivity are the exact same thing. However, precision and specificity are not the same thing. Precision makes sure that most of the predicted positives are truly positive, and specificity checks that most of the true negatives have been detected.
- As we increase the threshold in a model, we decrease its sensitivity and increase its specificity.
- The ROC, or receiver operating characteristic curve, is a useful graph that helps us keep track of the sensitivity and specificity of the model for each different value of the threshold.
- The ROC also helps us determine how good a model is, using the area under the curve, or AUC. The closer the AUC is to 1, the better the model. The closer the AUC is to 0.5, the worse the model.
- By looking at the ROC curve, we can make decisions on what threshold to use to give us good values for both the sensitivity

and the specificity, depending on how much of each our model expects. This makes the ROC curve one of the most popular and useful ways to evaluate and improve a model.

Exercises

EXERCISE 7.1

A video site has established that a particular user likes animal videos and absolutely nothing else. In the next figure, we can see the recommendations that this user got when logging in to the site.



If this is all the data we have on the model, answer the following questions:

a. What is the accuracy of the model?
b. What is the recall of the model?
c. What is the precision of the model?
d. What is the F_1-score of the model?
e. Would you say that this is a good recommendation model?

EXERCISE 7.2

Find the sensitivity and specificity of the medical model with the following confusion matrix:

	Predicted sick	Predicted healthy
Sick	120	22
Healthy	63	795

EXERCISE 7.3

For the following models, determine which error is worse, a false positive or a false negative. Based on that, determine which of the two metrics, precision or recall, we should emphasize on when evaluating each of the models.

1. A movie-recommendation system that predicts whether a user will watch a movie
2. An image-detection model used in self-driving cars that detects whether an image contains a pedestrian
3. A voice assistant at home that predicts whether the user gave it an order

EXERCISE 7.4

We are given the following models:

1. A self-driving car model for detecting a pedestrian based on the image from the car's camera
2. A medical model for diagnosing a deadly illness based on the patient's symptoms
3. A recommendation system for movies based on the user's previous movies watched

4. A voice assistant that determines whether the user needs assistance given the voice command
5. A spam-detection model that determines whether an email is spam based on the words in the email

We are given the task of evaluating these models using F_β-scores. However, we haven't been given the values of β to use. What value of β would you use to evaluate each of the models?

review on amazon

Up next...
8 Using probability to its maximum: The naive Bayes model
- what is Bayes theorem
- dependent and independent events
- the prior and posterior probabilities
- calculating conditional probabilities based on events
- using the naive Bayes model to predict whether an email is spam or ham, based on the words in the email
- coding the naive Bayes algorithm in Python

© 2024 Manning Publications Co.