Machine Learning for Biodiversity Scientists

An opinionated primer

Timothée Poisot

2024-11-21

Table of contents

List of Figures

Preface

Machine learning is now an established methodology to study biodiversity, and this is a problem.

This may be an opportunity when it comes to advancing our knowledge of biodiversity, and in particular when it comes to translating this knowledge into action (Tuia *et al.* 2022); but make no mistake, this is a problem for us, biodiversity scientists, as we suddenly need to develop competences in an entirely new field in order to remain professionally relevant (Ellwood *et al.* 2019). And as luck would have it, there are easier fields to master than machine learning. The point of this book, therefore, is to provide an introduction to fundamental concepts in data science, from the perspective of a biodiversity scientist, by using examples corresponding to real-world use-cases of these techniques.

But what do we mean by *machine learning* and *data science*? Most science, after all, relies on data in some capacity. What falls under the umbrella of *data science* is, in short, embracing in equal measure quantitative skills (mathematics, machine learning, statistics), programming, and domain expertise, in order to solve well-defined problems. *Machine learning* is a series of techniques (or, more precisely, a high-level approach to these techniques) through which we conduct our data science activities. A core tenet of data science is that, when using it, we seek to "deliver actionable insights", which is MBA-speak for "figuring out what to do next". One of the ways in which this occurs is by letting the data speak, after they have been, of course, properly cleaned and transformed and engineered. This entire process is driven by (or, even, subject to) domain knowledge. There is no such thing as data science, at least not in a vacuum: there is data science as a methodology applied to a specific domain.

Think of data science as being its own epistemology (Desai *et al.* 2022), and machine learning as one methodology we can apply to work within this context.

Before we embark into a journey of discovery on the applications of data science to biodiversity, allow me to let you in on a little secret: data *science* is a little bit of a misnomer. In order to understand why, I need (or at least, I really want) to talk about cooking.

To become a good cook, there are general techniques one *must* master, which we apply to specific steps in recipes; these recipes draw from a common cultural or local repertoire and cultural specifics (but the evolution of recipes is remarkably convergent – most cuisines have a *mirepoix*, bread, and beer). Finally, there is the product, *i.e.* the unique dish that you have cooked. And so it is for data science too: we can abstract a series of processes and guidelines, think about their application within the context of our specific field, study system, or line and research, and all of this will shape the final data product we can serve.

When writing this preface, I turned to my shelf of cookbooks, and picked my two favorites: Robuchon's *The Complete Robuchon* (a no-nonsense list of hundreds of recipes with no place for improvisation), and Bianco's *Pizza, Pasta, and Other Food I Like* (a short volume with very few pizza and pasta, and wonderful discussions about the importance of humility, creativity, and generosity). Data science, if it were cooking, would feel a lot like the second. Deviation from the rules is often justifiable if you feel like it. But this improvisation requires good skills, a clear mental map of the problem, a defined vision of what these deviations will let you achieve, and a library of patterns that you can draw from.

This book will not get you here. But it will speed up the process, by framing the practice of data science as a natural way to conduct research on biodiversity.

1. Introduction

This book started as a collection of notes from several classes I taught in the Department of Biological Sciences at the Université de Montréal, as well as a few workshops I ran for the Québec Centre for Biodiversity Sciences. When teaching data synthesis, data science, and machine learning to biology students, I realized that the field was missing resources that could serve as stepping stones to proficiency.

There are excellent manuals covering the mathematics of data science and machine learning (I will list a few later on). These are important to read, because the field of machine learning is an offshoot of mathematics and computer science, and it is important to become familiar with the core concepts. A little bit of calculus and a whole lot of linear algebra should be more of the same for many ecologists. But these resources are usually less useful as practical guides to the field.

There are many good papers giving overviews of some applications of data science to biological problems (a lot of them are cited in this book). These are important to read, because any attempt to adopt a new methodology (new to us, not new to the field, or new in absolute terms!) must proceed alongside some familiarity of how it has been used by our colleagues. But these articles, although good at showing how these tools are actually used, usually make it difficult to establish more general recommendations.

There are, finally, thousands of tutorials about how to write code to perform any machine learning algorithm you can think of. Some of them are even good. But these tutorials usually suffer (in our case) from being disconnected from the field of biodiversity science, and of course are limited by the language they use, the version of the packages they ran with, and again do not allow for much generalization.

When navigating these resources, one thing that students commonly called for was an attempt to tie concepts together, and to explain when and how human decisions were required in ML approaches (Sulmont *et al.* 2019).

This is particularly true of students with strong domain knowledge that want to understand how machine learning fits with their ability to do research.

This is book is this attempt.

There are, broadly speaking, two situations in which reading this book is useful. The first is when you are done reading some general books about machine learning, and want to see how it can be applied to problems that are more specific to biodiversity research; the second is when you have a working understanding of biodiversity research, and want a stepping stone into the machine learning literature. Note that there is no scenario where you *stop* after reading this book – this is by design. The purpose of this book is to give a practical overview of "how data science for biodiversity happens", and this needs to be done in parallel to even more fundamental readings.

These are examples of books I like. I found them comprehensive and engaging. They may not work for you.

A wonderful introduction to the mathematics behind machine learning can be found in Deisenroth *et al.* (2020), which provides stunning visualization of mathematical concepts. Yau (2015) is a particularly useful book about the ways to visualize data in a meaningful way. Watt *et al.* (2020) is a solid introduction to the underlying theory of applied machine learning. For ecologists, Dietze (2017) is a comprehensive, and still highly readable, treaty on the problems associated to forecasting. The best way to decide on which book to read is often to look at the books that your colleagues have also read; being able to work through material collectively is useful, and knowing that you can practice the craft of data science within a community will make your learning more effective.

When reading this book, I encourage you to read the chapters in order. They have been designed to be read in order, because each chapter introduces the least possible amount of new concepts, but often requires to build on the previous chapters. This is particularly true of the second half of this book.

1.1. Core concepts in data science

1.1.1. EDA

1.1.2. Clustering and regression

1.1.3. Supervised and unsupervised

1.1.4. Training, testing, and validation

1.1.5. Transformations and feature engineering

1.2. An overview of the content

In **?@sec-clustering**, we introduce some fundamental questions in data science, by working on the clustering of pixels in Landsat data. The point of this chapter is to question the way we think about data, and to start a discussion about an "optimal" model, hyper-parameters, and what a "good" model is.

In **?@sec-gradientdescent**, we revisit well-trodden statistical ground, by fitting a linear model to linear data, but uisng gradient descent. This provides us with an opportunity to think about what a "fitted" model is, whether it is possible to learn too much from data, and why being able to think about predictions in the unit of our problem helps.

In **?@sec-crossvalidation**, we start introducing one of the most important bit element of data science practice, in the form of cross-validation. We apply this technique to the prediction of plant phenology over a millenia, and think about the central question of "what kind of decision-making can we justify with a model".

In Chapter 2, we introduce the task of classification, and spend a lot of time thinking about biases in predictions, which are acceptable, and which are not. We start building a model for the distribution of the Reindeer, which we will improve over a few chapters.

In Chapter 3, we explore ways to perform variable selection, think of this task as being part of the training process, and introduce ideas related to dimensionality reduction. In Section 3.2, we discuss data leakage, where it comes from, and how to prevent it. This leads us to introducing the concept of data transformations as a model, which will establish some best practices we will keep on using throughout this book.

In Chapter 4, we conclude story arcs that had been initiated in a few previous chapters, and explore training curves, the tuning of hyper-parameters, and moving-threshold classification. We provide the final refinements to out model of the Reindeer distribution.

In **?@sec-explanations**, we will shift our attention from prediction to understanding, and explore techniques to quantify the importance of variables, as well as ways to visualize their contribution to the predictions. In doing so, we will introduce concepts of model interpretation and explainability.

In **?@sec-bagging**, ...

1.3. A note on colors

Type	Meaning	Color
All	generic	
	no data	
Cross-validation	training	
	validation	

Type	Meaning	Color
Species range	testing	
	presence	
	absence	
Range change	loss	
	no change	
	gain	

In addition, there are three important color *palettes*. Information that is *sequential* is nature, which is to say it increases on a continuous scale without a logical midpoint, is rendered with these colors (from low to the left, to high values to the right):

The diverging palette is used for values that have a clear midpoint (usually values centered on 0). The midpoint will always correspond to the central color, and this palette is symmetrical:

Finally, the categorical data are represented using the following palette:

1.4. Some rules about this book

When I started aggregating these notes, I decided on a series of four rules. No code, no simulated data, no long list of model, and above all, no *iris* dataset. In this section, I will go through *why* I decided to adopt these rules, and how it should change the way you interact with the book.

1.4.1. No code

This is, maybe, the most surprising rule, because data science *is* programming (in a sense). But sometimes there is so much focus on programming that we lose track of the other, important aspects of the practice of data science: abstractions, relationship with data, and domain knowledge.

This book *did* involve a lot of code. Specifically, this book was written using *Julia* (Bezanson *et al.* 2017), and every figure is generated by a notebook, and they are part of the material I use when teaching from this content in the classroom. But code is *not* a universal language, and unless you are really familiar with the language, code can obfuscate. I had no intention to write a *Julia* book (or an *R* book, or a *Python* book). The point is to

think about data science applied to ecological research, and I felt like it would be more inclusive to do this in a language agnostic way.

And finally, code rots. Code with more dependencies rots faster. It take a single change in the API of a package to break the examples, and then you are left with a very expensive monitor stand. With a few exceptions, the examples in this book do not use complicated packages either.

1.4.2. No simulated data

I have nothing against simulated data. I have, in fact, generated simulated data in many different contexts, for training or for research. But the limit of simulated is that we almost inevitably fail to include what makes real data challenging: noise, incomplete or uneven sampling, data representation artifacts. And so when it is time to work on real data, everything seems suddenly more difficult.

Simulated data have *immense* training value; but it is also important to engage with the imperfect actual data, as we will overwhelmingly apply the concepts from this book to them. For this reason, there are no simulated data in this book. Everything that is presented correspond to an actual use case that proceeds from a question we could reasonably ask in the context, paired with a dataset that could be used to answer this question.

1.4.3. No model zoo

My favorite machine learning package is *MLJ* (Blaom *et al.* 2020). When given a table of labels and a table of features, it will give back a series of models that match with these data. It speeds up the discovery of models considerably, and is generally a lot more informative than trying to read from a list of possible techniques. If I have questions about an algorithm from this list, I can start reading more documentation about how it works.

Reading a long enumeration of things is boring; unless it's sung by Yakko Warner, I'm not interested, and I refuse to inflict it on people. But more importantly, these enumerations of models often distract from thinking about the problem we want to solve in more abstract terms. I rarely wake up in the morning and think "oh boy I can't wait to train a SVM today"; chances are, my thought process will be closer to "I need to tell the mushroom people where I think the next good foraging locations will be". The rest, is implementation details.

In fact, 90% of this book uses only two models: linear regression, and the Naïve Bayes Classifier. Some other models are involved in a few chapters, but these two models are breathtakingly simple, work surprisingly well, run fast, and can be tweaked to allow us to build deep intuitions about how machines learn. They are perfect for the classroom, and give us the freedom to spent most of our time thinking about how we interact with models, and why, and how we make methodological decisions.

1.4.4. No *iris* dataset

From a teaching point of view, the *iris* dataset is like hearing Smash Mouth in a movie trailer, in that it tells you two things with absolute certainty. First, that you are indeed watching a movie trailer. Second, that you could be watching Shrek instead. There are datasets out there that are *infinitely more* exciting to use than *iris*.

But there is a far more important reason not to use *iris*: eugenics.

Listen, we made it several hundred words in a text about quantitative techniques in life sciences without encountering a sad little man with racist ideas that academia decided to ignore because "he just contributed so much to the field, and these were different times, maybe we shouldn't be so quick to judge?". Ronald Aylmer Fisher, statistics' most racist nerd, was such a man; and there are, of course, those who want to consider the possibility that you can be outrageously racist as long as you are an outstanding scientist (Bodmer *et al.* 2021).

The *iris* dataset was first published by Fisher (1936) in the *Annals of Eugenics* (so, there's a bit of a red flag there already), and draws from several publications by Edgar Anderson, starting with Anderson (1928); Unwin & Kleinman (2021) have an interesting historiographic deep-dive into the correspondence between the two. Judging by the dates, you may think that Fisher was a product of his time. But this could not be further from the truth. Fisher was dissatisfied with his time, to the point where his contributions to statistics were done in service of his views, in order to provide the appearance of scientific rigor to his bigotry.

Fisher advocated for forced sterilization for the "defectives" (which he estimated at, oh, roughly 10% of the population), argued that not all races had equal capacity for intellectual and emotional development, and held a host of related opinions. There is no amount of contribution to science that pardon these views. Coming up with the idea of the null hypothesis does not even out lending "scientific" credibility to ideas whose logical

(and historical) conclusion is genocide. That Ronald Fisher is still described as a polymath and a genius is infuriating, and we should use every alternative to his work that we have.

Thankfully, there are alternatives!

The most broadly known alternative to the *iris* dataset is `penguins`, which was collected by ecologists (Gorman *et al.* 2014), and published as a standard dataset (Horst *et al.* 2020) so that we can train students without engaging with the "legacy" of eugenicists. The `penguins` dataset is also genuinely good! The classes are not so obviously separable, there are some missing data that reflect the reality of field work, and the data about sex and spatial location have been preserved, which increases the diversity of questions we can ask. We won't use `penguins` either. It's a fine dataset, but at this point there is little that we can write around it that would be new, or exciting. But if you want to apply some of the techniques in this book? Go `penguins`.

References

Anderson, E. (1928). The problem of species in the northern blue flags, iris versicolor l. And iris virginica l. *Annals of the Missouri Botanical Garden*, 15, 241.

Bezanson, J., Edelman, A., Karpinski, S. & Shah, V.B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59, 65–98.

Blaom, A., Kiraly, F., Lienart, T., Simillides, Y., Arenas, D. & Vollmer, S. (2020). MLJ: A julia package for composable machine learning. *Journal of Open Source Software*, 5, 2704.

Bodmer, W., Bailey, R.A., Charlesworth, B., Eyre-Walker, A., Farewell, V., Mead, A., *et al.* (2021). The outstanding scientist, R.A. Fisher: his views on eugenics and race. *Heredity*, 126, 565–576.

Deisenroth, M.P., Faisal, A.A. & Ong, C.S. (2020). Mathematics for machine learning.

Dietze, M. (2017). Ecological forecasting.

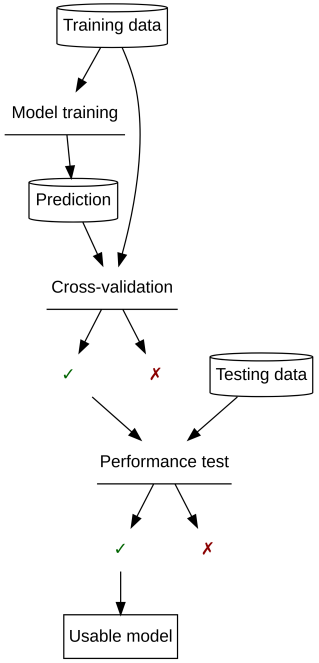
Fisher, R.A. (1936). The Use Of Multiple Measurements In Taxonomic Problems. *Annals of Eugenics*, 7, 179–188.

Gorman, K.B., Williams, T.D. & Fraser, W.R. (2014). Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis). *PLoS ONE*, 9, e90081.

Horst, A.M., Hill, A.P. & Gorman, K.B. (2020). *Allisonhorst/palmerpenguins: v0.1.0*. Zenodo.

Sulmont, E., Patitsas, E. & Cooperstock, J.R. (2019). Can you teach me to machine learn? *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*.

Unwin, A. & Kleinman, K. (2021). The Iris Data Set: In Search of the Source of *Virginica*. *Significance*, 18, 26–29.
Watt, J., Borhani, R. & Katsaggelos, A. (2020). Machine learning refined.
Yau, N. (2015). Visualize this.



Flowchart 1.1: An overview of the process of coming up with a usable model. The process of creating a model starts with a trainig dataset made of predictors and responses, which is used to train a model. This model is cross-validated on its training data, to estimate whether it can be fully retrained. The fully trained model is that applied to an independent testing dataset, and the evaluation of the performance determines whether it will be used.

2. Supervised classification

In the previous chapters, we have focused our efforts on regression models, which is to say models that predict a continuous response. In this chapter, we will introduce the notion of classification, which is the prediction of a discrete variable representing a category. There are a lot of topics we need to cover before we can confidently come up with a model for classification, and so this chapter is part of a series. We will first introduce the idea of classification; in Chapter 3, we will explore techniques to fine-tune the set of variables we use for prediction; in Chapter 4, we will think about predictions of classes as probabilities, and generalize these ideas and think about learning curves; finally, in **?@sec-explanations**, we will think about variables a lot more, and introduce elements of model interpretability.

2.1. The problem: distribution of an endemic species

Throughout these chapters, we will be working on a single problem, which is to predict the distribution of the Corsican nuthatch, *Sitta whiteheadi*. The Corsican nuthatch is endemic to Corsica, and its range has been steadily shrinking over time due to loss of habitat through human activity, including fire, leading to it being classified as "vulnerable to extinction" by the International Union for the Conservation of Nature. Barbet-Massin & Jiguet (2011) nevertheless show that the future of this species is not necessarily all gloom and doom, as climate change is not expected to massively affect its distribution.

Species Distribution Modeling (SDM; Elith & Leathwick (2009)), also known as Ecological Niche Modeling (ENM), is an excellent instance of ecologists doing applied machine learning already, as Beery *et al.* (2021) rightfully pointed out. In fact, the question of fitness-for-purpose, which we discussed in previous chapters (for example in **?@sec-crossvalidation-fitness**), has been covered in the SDM literature (Guillera-Arroita *et al.*

2015). In these chapters, we will fully embrace this idea, and look at the problem of predicting where species can be as a data science problem. In the next chapters, we will converge again on this problem as an ecological one. Being serious about our data science practices when training a species distribution model is important: Chollet Ramampiandra *et al.* (2023) make the important point that it is easy to overfit more complex models, at which point they cease outperforming simple statistical models.

Because this chapter is the first of a series, we will start by building a bare-bones model on ecological first principles. This is an important step. The rough outline of a model is often indicative of how difficult the process of training a really good model will be. But building a good model is an iterative process, and so we will start with a very simple model and training strategy, and refine it over time. In this chapter, the purpose is less to have a very good training process; it is to familiarize ourselves with the task of classification.

We will therefore start with a blanket assumption: the distribution of species is something we can predict based on temperature and precipitation. We know this to be important for plants and animals (Clapham *et al.* 1935; Whittaker 1962), to the point where the relationship between mean temperature and annual precipitation is how we find delimitations between biomes. If you need to train a lot of models on a lot of species, temperature and precipitation are not the worst place to start (Berteaux 2014).

Consider our dataset for a minute. In order to predict the presence of a species, we need information about where the species has been observed; this we can get from the Global Biodiversity Information Facility. We need information about where the species has *not* been observed; this is usually not directly available, but there are ways to generate background points that are a good approximation of this (Barbet-Massin *et al.* 2012; Hanberry *et al.* 2012). All of these data points come in the form (lat., lon., y), which give a position in space, as well as $y = \{+, -\}$ (the species is present or absent!) at this position.

To build a model with temperature and precipitation as inputs, we need to extract the temperature and precipitation at all of these coordinates. We will use the CHELSA1 dataset (Karger *et al.* 2017), at a resolution of 30 seconds of arc. WorldClim2 (Fick & Hijmans 2017) also offers access to similar bioclimatic variables, but is known to have some artifacts that may bias the analysis.

The predictive task we want to complete is to get a predicted presence or absence $\hat{y} = \{+, -\}$, from a vector $\mathbf{x}^\top = [\text{temp.} \quad \text{precip.}]$. This specific task is called classification, and we will now introduce some elements of theory.

2.2. What is classification?

Classification is the prediction of a qualitative response. In **?@sec-clustering**, for example, we predicted the class of a pixel, which is a qualitative variable with levels $\{1, 2, \dots, k\}$. This represented an instance of *unsupervised* learning, as we had no *a priori* notion of the correct class of the pixel. When building SDMs, by contrast, we often know where species are, and we can simulate "background points", that represent assumptions about where the species are not. For this series of chapters, the background points have been generated by sampling preferentially the pixels that are farther away from known presences of the species.

In short, our response variable has levels $\{+, -\}$: the species is there, or it is not – we will challenge this assumption later in the series of chapters, but for now, this will do. The case where the species is present is called the *positive class*, and the case where it is absent is the *negative class*. We tend to have really strong assumptions about classification already. For example, monitoring techniques using environmental DNA (*e.g.* Perl *et al.* 2022) are a classification problem: the species can be present or not, $y = \{+, -\}$, and the test can be positive of negative $\hat{y} = \{+, -\}$. We would be happy in this situation whenever $\hat{y} = y$, as it means that the test we use has diagnostic value. This is the essence of classification, and everything that follows is more precise ways to capture how close a test comes from this ideal scenario.

When working on $\{+, -\}$ outcomes, we are specifically performing *binary* classification. Classification can be applied to more than two levels.

2.2.1. Separability

A very important feature of the relationship between the features and the classes is that, broadly speaking, classification is much easier when the classes are separable. Separability (often linear separability) is achieved when, if looking at some projection of the data on two dimensions, you can draw a line that separates the classes (a point in a single dimension, a plane in three dimension, and so on and so forth). For reasons that will become clear in Section 3.3.1, simply adding more predictors is not the right thing to do.

In Figure 2.1, we can see the temperature (in degrees) for locations with recorded presences of Corsican nuthatches, and for locations with assumed absences. These two classes are not quite linearly separable alongside this single dimension (maybe there is a different projection of the data that would change this; we will explore one in Chapter 3), but there are still some values at which our guess for a class changes. For example, at a location with a temperature colder than 10°C, presences are far more likely. For a location with a temperature

warmer than 15°C, absences become overwhelmingly more likely. The locations with a temperature between 10°C and 15°C can go either way.

2.2.2. The confusion table

Evaluating the performance of a classifier (a classifier is a model that performs classification) is usually done by looking at its confusion table, which is a contingency table of the form

$$\begin{pmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{pmatrix}. \tag{2.1}$$

This can be stated as "counting the number of times each pair of (prediction, observation occurs)", like so:

$$\begin{pmatrix} |\hat{+}, +| & |\hat{+}, -| \\ |\hat{-}, +| & |\hat{-}, -| \end{pmatrix}. \tag{2.2}$$

The four components of the confusion table are the true positives (TP; correct prediction of $+$), the true negatives (TN; correct prediction of $-$), the false positives (FP; incorrect prediction of $+$), and the false negatives (FN; incorrect prediction of $-$). Quite intuitively, we would like our classifier to return mostly elements in TP and TN: a good classifier has most elements on the diagonal, and off-diagonal elements as close to zero as possible (the proportion of predictions on the diagonal is called the accuracy, and we will spend Section 2.2.4 discussing why it is not such a great measure).

As there are many different possible measures on this matrix, we will introduce them as we go. In this section, it it more important to understand how the matrix responds to two important features of the data and the model: balance and bias.

Balance refers to the proportion of the positive class. Whenever this balance is not equal to $1/2$ (there are as many positives as negative cases), we are performing *imbalanced* classification, which comes with additional challenges; few ecological problems are balanced.

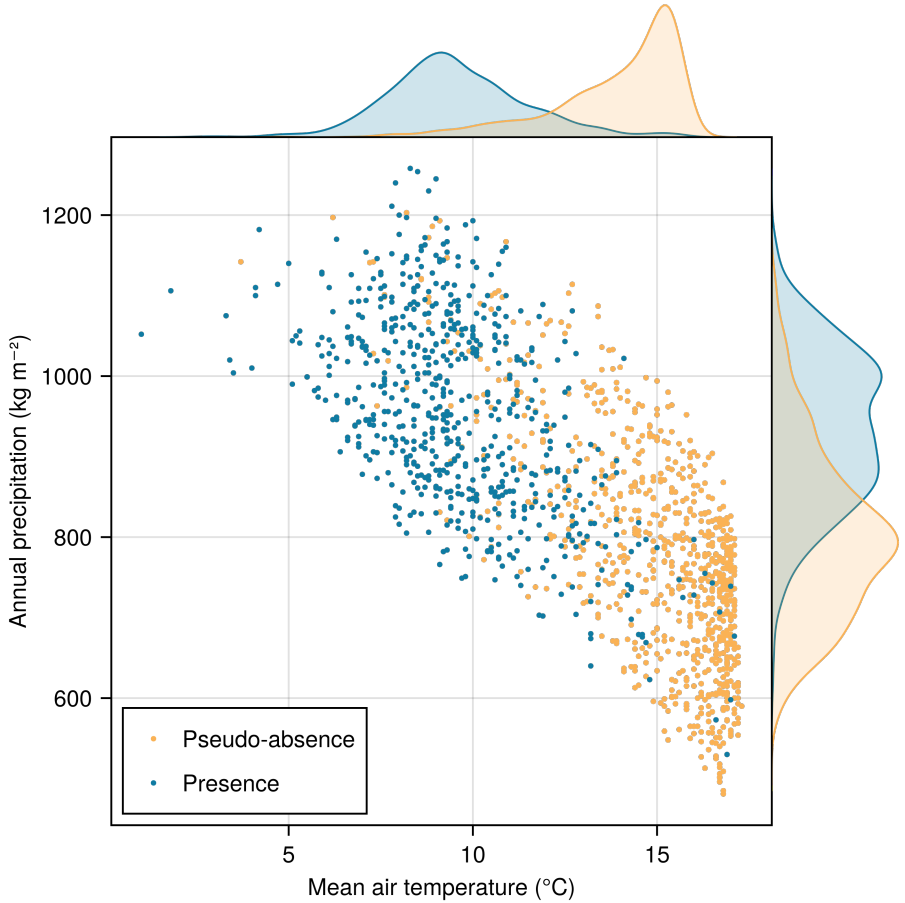


Figure 2.1.: This figures show the separability of the presences and pseudo-absences on the temperature and precipitation dimensions.

2.2.3. Null classifiers

A useful baseline to establish whether a model "works" is to measure whether the model performs better than at random. For classification problems, a good baseline is provided by "null" classifiers, in which the underlying structure of the data is known (and respects class balance), but the classifier itself makes guesses at random, for different definitions of random. Because these classifiers are very simple, they are not in fact models; we can directly write their confusion matrix, and apply different measures of model performance to it.

2.2.3.1. The no-skill classifier

There is a specific hypothetical classifier, called the *no-skill classifier*, which guesses classes at random as a function of their proportion. It turns out to have an interesting confusion matrix! If we note b the proportion of positive classes, the no-skill classifier will guess + with probability b, and − with probability $1 - b$. Because these are also the proportion in the data, we can write the confusion matrix as

$$\begin{pmatrix} b^2 & b(1-b) \\ (1-b)b & (1-b)^2 \end{pmatrix}. \tag{2.3}$$

The proportion of elements that are on the diagonal of this matrix is $b^2 + (1-b)^2$. When b gets lower, this value actually increases: the more difficult a classification problem is, the more accurate random guesses *look like*. This has a simple explanation, which we expand Section 2.2.4 : when most of the cases are negative, if you predict a negative case often, you will by chance get a very high true negative score. For this reason, measures of model performance will combine the positions of the confusion table to avoid some of these artifacts.

Bias refers to the fact that a model can recommend more (or fewer) positive or negative classes than it should. An extreme example is the *zero-rate classifier*, which will always guess the most common class, and which is commonly used as a baseline for imbalanced classification. A good classifier has high skill (which we can measure by whether it beats the no-skill classifier for our specific problem) and low bias. In this chapter, we will explore different measures on the confusion table the inform us about these aspects of model performance, using the Naive Bayes Classifier.

2.2.3.2. The coin-flip classifier

An alternative to the no-skill classifier is the coin-flip classifier, in which classes have their correct prevalence b, but the model picks at random with probability $1/2$ within these classes. This differs from the no-skill classifier by adopting a different random chance of picking a class while still respecting the prevalence of the positive class.

The confusion matrix of the coin-flip classifier is:

$$\begin{pmatrix} \frac{b}{2} & \frac{1-b}{2} \\ \frac{b}{2} & \frac{1-b}{2} \end{pmatrix}.$$

2.2.3.3. The constant classifier

The last null classifier we can use is the constant classifier, in which we assume that the model will *always* return some specific class. This is useful to anticipate what the model performance would look like if, for example, the model always predicted the negative outcome (which is a notion we will return to in Section 2.2.4).

This classifier has the confusion matrix

$$\begin{pmatrix} b & 1-b \\ 0 & 0 \end{pmatrix}$$

if it always predicts the positive class, and

$$\begin{pmatrix} 0 & 0 \\ b & 1-b \end{pmatrix}$$

if it always predicts the negative class.

2.2.4. A note on accuracy

It is tempting to use accuracy to measure how good a classifier is, because it makes sense: it quantifies how many predictions are correct. But a good accuracy can hide a very poor performance. Let's think about an extreme case, in which we want to detect an event that happens with prevalence 0.05. Out of 100 predictions, the confusion matrix of this model would be

$$\begin{pmatrix} 0 & 0 \\ 5 & 95 \end{pmatrix}.$$

The accuracy of this classifier would be 0.95, which seems extremely high! This is because prevalence is extremely low, and so most of the predictions are about the negative class: the model is *on average* really good, but is completely missing the point when it comes to making interesting predictions.

In fact, even a classifier that would not be that extreme would be mis-represented if all we cared about was the accuracy. If we take the case of the no-skill classifier, the accuracy is given by $b^2 + (1-b)^2$, which is an inverted parabola that is *maximized* for $b \approx 0$ – a model guessing at random will appear better when the problem we want to solve gets more difficult. This effect is called the paradox of accuracy.

Whenever possible, avoid using accuracy except to communicate the skill of the model in easy to understand terms.

This is an issue inherent to accuracy: it can tell you that a classifier is bad (when it is low), but it cannot really tell you when a classifier is *good*, as no-skill (or worse-than-no-skill) classifiers can have very high values. It remains informative as an *a posteriori* measure of performance, but only after using reliable measures to ensure that the model means something.

More generally, this also illustrates why relying on null classifiers is a good idea: we want to make sure that we are making better predictions than an heavily biased, uninformative model would.

2.3. The Naive Bayes Classifier

In practice, we do not use the Naive Bayes Classifier for SDMs. There are far more powerful alternatives based on boosting, like boosted regression trees, or Bayesian additive regression trees. But NBC makes for an easy to follow example across many chapters.

The Naive Bayes Classifier (NBC) is my all-time favorite classifier. It is built on a very simple intuition, works with almost no data, and more importantly, often provides an annoyingly good baseline for other, more complex classifiers to meet. That NBC works *at all* is counter-intuitive (Hand & Yu 2001). It assumes that all variables are independent, it works when reducing the data to a simpler distribution, and although the numerical estimate of the class probability can be somewhat unstable, it generally gives good predictions. NBC is the data science equivalent of saying "eh, I reckon it's probably *this* class" and somehow getting it right 95% of the time. There are, in fact, several papers questioning *why* NBC works *at all* (see *e.g.* Kupervasser 2014).

2.3.1. How the NBC works

In Figure 2.1, what is the most likely class if the temperature is 8°C?

We can look at the density traces on top, and say that because the one for presences is higher, we would be justified in guessing that the species is present. Of course, this is equivalent to saying that $P(8°C|+) > P(8°C|-)$. It would appear that we are looking at the problem in the wrong way, because we are really interested in $P(+|8°C)$, the probability that the species is present knowing that the temperature is 8°C.

Using Baye's theorem, we can re-write our goal as

$$P(+|x) = \frac{P(+)}{P(x)} P(x|+)\,, \tag{2.4}$$

where x is one value of one feature, $P(x)$ is the probability of this observation (the evidence, in Bayesian parlance), and $P(+)$ is the probability of the positive class (in other words, the prior). So, this is where the "Bayes" part comes from.

But why is NBC naive?

In Equation 2.4, we have used a single feature x, but the problem we want to solve uses a vector of features, $\mathbf{x}$. These features, statisticians will say, will have covariance, and a joint distribution, and many things that will challenge the simplicity of what we have written so far. These details, NBC says, are meaningless.

NBC is naive because it makes the assumptions that the features are all independent. This is actually the foundation upon which the NBC is built. To express the assumption of features independence, we simply need to write that $P(+|\mathbf{x}) \propto P(+) \prod_i P(\mathbf{x}_i|+)$ (by the chain rule). Note that this is not a strict equality: to get the actual value of $P(+|\mathbf{x})$ we need to divide by the evidence, and so we need to find the expression of the evidence. But instead of doing this, we simply have to note that the evidence is constant across all classes, and so we do not need to measure it to get an estimate of the score for a class. We can think of this assumption in a problem-specific way: if we walk across a landscape at random with regard to our response variable, *i.e.* we do not know whether the species will be present or not, there is (i) no reason to assume that the probability of measuring a specific temperature (or other feature) will be linked to the response in any way, and (ii) no reason to assume that a third, mysterious value that is neither presence nor absence could ever be measured; therefore, $P(\mathbf{x})$ is a constant for our model.

To generalize our notation, the score for a class $\mathbf{c}_j$ is $P(\mathbf{c}_j) \prod_i P(\mathbf{x}_i|\mathbf{c}_j)$. In order to decide on a class, we apply the following rule:

$$\hat{y} = \text{argmax}_j P(\mathbf{c}_j) \prod_i P(\mathbf{x}_i|\mathbf{c}_j)\,. \tag{2.5}$$

In other words, whichever class gives the higher score, is what the NBC will recommend for this instance $\mathbf{x}$. In Chapter 4, we will improve upon this model by thinking about (and eventually calculating) the evidence $P(\mathbf{x})$ in order to estimate the actual probability, but as you will see, this simple formulation will already prove frightfully effective.

2.3.2. How the NBC learns

There are two unknown quantities at this point. The first is the value of $P(+)$ and $P(-)$. These are priors, and are presumably important to pick correctly. In the spirit of iterating rapidly on a model, we can use two starting points: either we assume that the classes have the same probability, or we assume that the representation of the classes (the balance of the problem) *is* their prior. It now helps to think about the no-skill and coin-flip classifier we introduced earlier in the chapter. Assume that we do not use $P(x|c)$ when making our prediction: the baseline against which we compare the model will therefore be entire determined by $P(+)$. Picking a prior

of one half is making the predictions at random (like coin-flip), and picking a prior equal to the prevalence is making the predictions at random (like no-skill). Understanding how we set the prior for the NBC is important, as it can ensure that we use a fair baseline to compare it against. Throughout this book, we will let our prior be the prevalence in the training data (and therefore our first task will be to beat the no-skill classifier). Finally, note that we do not need to think about $P(-)$ too much, as it is simply $1 - P(+)$: the "state" of every single observation of the presence or absence of the species under a set of measured environmental variables is either $+$ or $-$.

The most delicate problem is to figure out $P(x|c)$, the probability of the observation of the variable when the class is known. There are variants here that will depend on the type of data that is in x; as we work with continuous variables, we will rely on Gaussian NBC. In Gaussian NBC, we will consider that x comes from a normal distribution $\mathcal{N}(\mu_{x,c}, \sigma_{x,c})$, and therefore we simply need to evaluate the probability density function of this distribution at the point x. Other types of data are handled in the same way, with the difference that they use a different set of distributions; for example, categorical variables can be represented using multinomial distributions (Abbood *et al.* 2020).

We could use different approaches to NBC, by using (for example) the empirical CDF function of the training data for each class. We will revisit this idea in chapter **?@sec-squint**, as it establishes an interesting parallel between different methods.

Therefore, the learning stage of NBC is extremely quick: we take the mean and standard deviation of the values, split by predictor and by class, and these are the parameters of our classifier. By contrast to the linear regression approach we worked with in **?@sec-gradientdescent**, the learning phase only involves a single epoch: measuring the mean and standard deviation. This yields a Gaussian NBC with the assumption that variables are normally distributed, because the normal distribution is the maximal entropy distribution when we know these two moments. This also reveals an interesting feature of NBC: it can work when we do not have access to the underlying training data. Imagine a situation where we only have access to published summary statistics about the environmental variables for which the species was observed / not observed: we can use these to establish the Normal distributions for each feature for each class, and use the NBC. Its ability to work under extreme data scarcity (assuming we are comfortable with the assumptions about the shape of the distribution) makes NBC a surprisingly versatile classifier.

2.4. Application: a baseline model of the Corsican nuthatch

In this section, we will have a look at the temperature and precipitation data from Figure 2.1, and come up with a first version of our classifier, which is to say: we will train our first attempt at an ecological niche model for the Corsican nuthatch.

2.4.1. Training and validation strategy

To evaluate our model, as we discussed in **?@sec-crossvalidation**, we will keep a holdout testing set, that will be composed of 20% of the observations. In this chapter, we will not be using these data, because in order to use them as a stand-in for future predictions, it is important that the model only sees them once (this will happen at the end of Chapter 4). Therefore, for the next chapters, we will limit ourselves to an evaluation of the model performance based on the average values of the performance measure we picked as the most informative, calculated on the validation datasets. When, based on this criteria, we have identified and validated the best model, we will evaluate it on the testing data.

In this chapter, we will rely on Monte-Carlo cross validation (MCCV; see **?@sec-crossvalidation-montecarlo**), using 100 replicates. In the following chapters, we will revert to using k-folds cross-validation, but using MCCV here is a good enough starting point.

In order to see how good our model really is, we will also compare its performances to various null classifiers. There are almost never difficult classifiers to outperform, but this nevertheless provides a good indication of whether our model works *at all*. In **?@sec-squint**, we will introduce a slightly more domain-specific model to provide a baseline that would look like an actual model we would like to out-perform (but mostly to make the general point that any problem can be approached like a machine learning problem).

2.4.2. Performance evaluation of the model

In order to get a sense of the performance of our model, we will need to decide on a performance measure. This is an important step, as we will use the average value of this measure on the validation data to decide

on the best model *before* reporting the expected performance. If we pick a measure that is biased, we will therefore use a model that is biased. Following Chicco & Jurman (2020) and Jurman *et al.* (2012), we will use the Matthew's Correlation Coefficient (MCC) as the "main" measure to evaluate the performance of a model (we will return to other alternative measures in Chapter 4, and eventually explain why MCC is the most appropriate for classification evaluation).

The MCC is defined as

$$\frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \times (\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TN} + \text{FN})}}.$$

The MCC is a correlation coefficient. Specifically, it is the Pearson product-moment correlation on a contingency table, where the contingency table is the confusion table (Powers 2020). Therefore, it returns values in $[-1, 1]$, which can be interpreted as every other correlation value. A negative value indicates perfectly wrong predictions, a value of 0 indicates no-skill, and a value of 1 indicates perfect predictions. By picking the model with the highest MCC on the validation data, we are likely to pick the best possible model (after controlling for over-fitting).

In addition to reporting the MCC, we will also look at values that inform us on the type of biases in the model, namely the positive and negative predictive values. These values, respectively TP/(TP+FP) and TN/(TN+FN), measure how likely a prediction of, respectively, presence and absence, are. To put it in other words, they measure how much the "true" events are represented in all of the predictions for a given even type: a PPV value of 0.7 means that 7 out of 10 positive predictions were true positives. These range in $[0, 1]$, and values of one indicate a better performance of the model. It may help to sometimes talk about the falses predictions, in which case the false omission rate (1 - NPV) and false discovery rate (1 - PPV) can be used: they quantify the *risk* we take when acting on a positive or negative recommendation from the model.

Why not pick one of these instead of the MCC? Because all modeling is compromise; we don't want a model to become too good at predicting absences, to the point where prediction about presences would become meaningless. Selecting models on the basis of a measure that only emphasizes one outcome is a risk that we shouldn't be willing to take. For this reason, measures that are good at optimizing the value of a negative and a positive prediction are far better representations of the performance of a model. The MCC does just this.

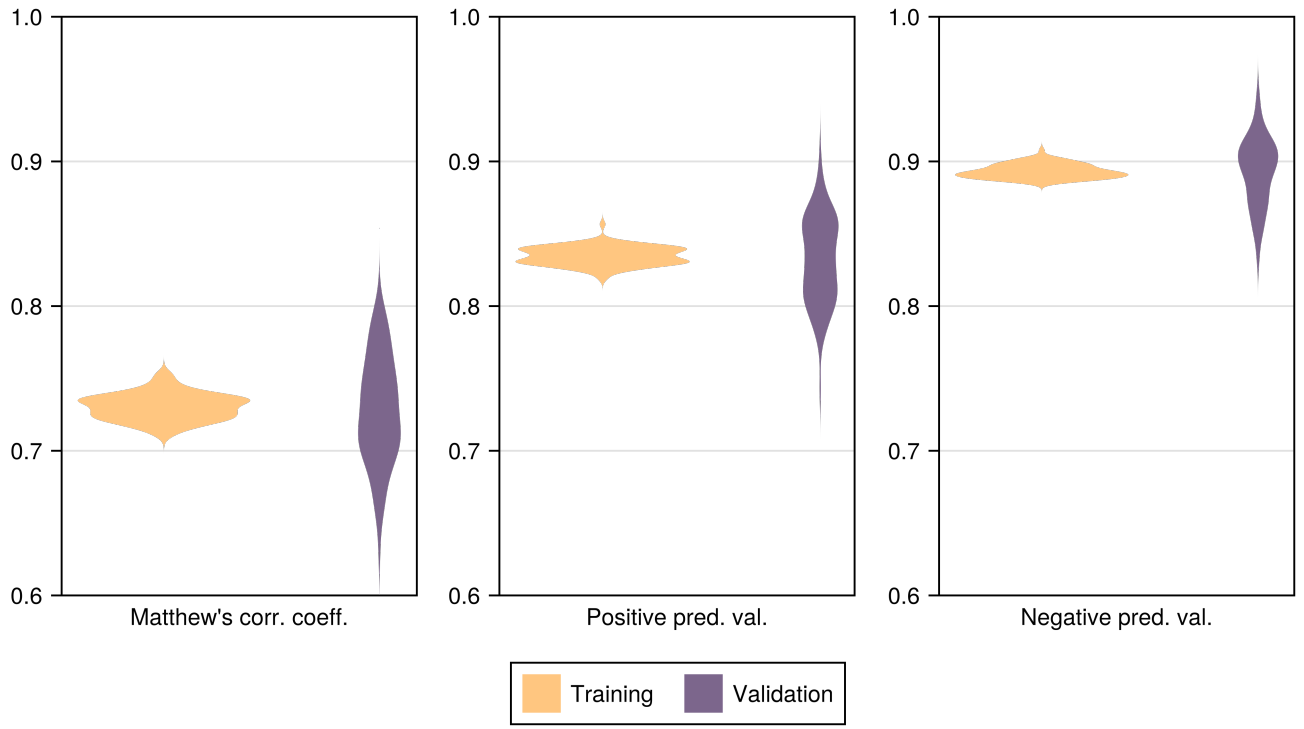


Figure 2.2.: Overview of the scores for the Matthew's correlation coefficient, as well as the positive and negative predictive values.

The output of cross-validation is given in Figure 2.2 (and compared to the no-skill classifier in Table 2.1). As we are satisfied with the model performance, we can re-train it using all the data (*but not the part used for testing*) in order to make our first series of predictions.

Table 2.1.: Overview of the data presented in Figure 2.2, compared to the null classifiers from Section 2.2.3. Note that the MCC gives values of 0 for *most* null classifiers, which is not the case with other measures of performance. Missing values cannot be calculated as they involved a denominator of 0.

Measure	**Training**	**Validation**	**No-skill**	**Coin-flip**	**Positive**	**Negative**
Accuracy	0.87	0.87	0.51	0.43	0.43	0.57
NPV	0.89	0.89	0.57	0.43		0.57
PPV	0.83	0.83	0.43	0.43	0.43	
MCC	0.73	0.73	-0.00	-0.15	0.00	0.00

2.4.3. The decision boundary

Now that the model is trained, we can take a break in our discussion of its performance, and think about *why* it makes a specific classification in the first place. Because we are using a model with only two input features, we can generate a grid of variables, and the ask, for every point on this grid, the classification made by our trained model. This will reveal the regions in the space of parameters where the model will conclude that the species is present.

The output of this simulation is given in Figure 2.3. Of course, in a model with more features, we would need to adapt our visualisations, but because we only use two features here, this image actually gives us a complete understanding of the model decision process. Think of it this way: even if we lose the code of the model, we could use this figure to classify any input made of a temperature and a precipitation, and read what the model decision would have been.

The line that separates the two classes is usually referred to as the "decision boundary" of the classifier: crossing this line by moving in the space of features will lead the model to predict another class at the output. In this

Take a minute to think about which places are more likely to have lower temperatures on an island. Is there an additional layer of geospatial information we could add that would be informative?

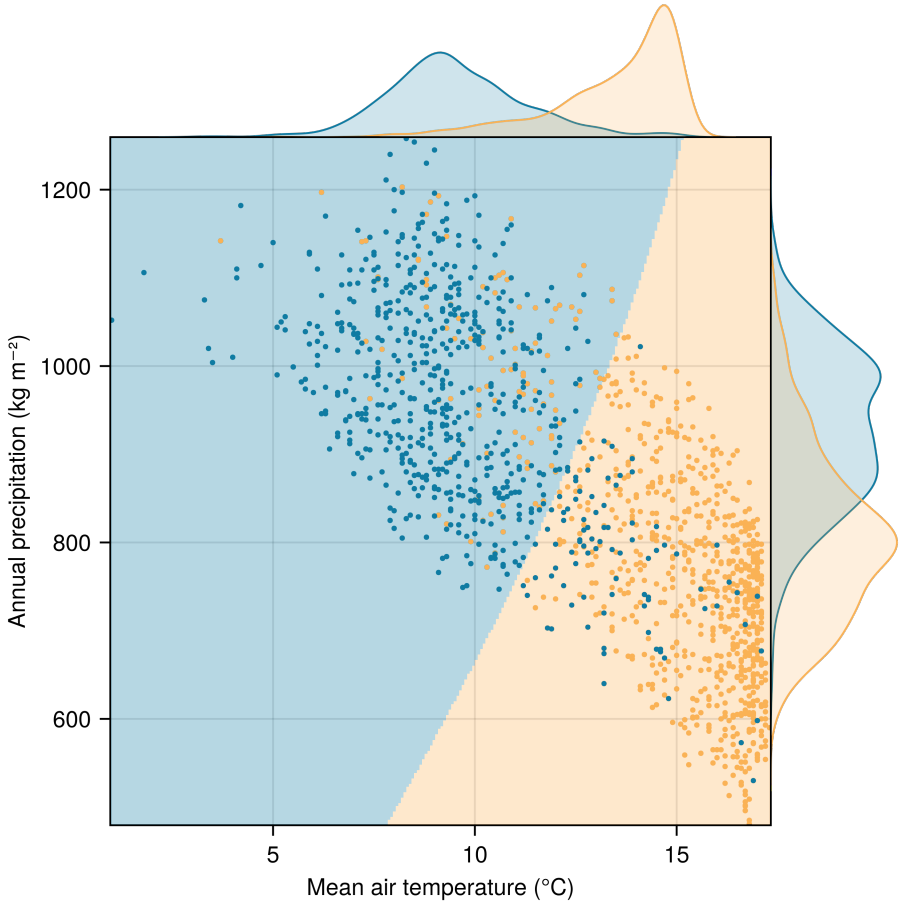


Figure 2.3.: Overview of the decision boundary between the positive (orange) and negative (grey) classes using the NBC with two variables. Note that, as expected with a Gaussian distribution, the limit between the two classes looks circular. The assumption of statistical independance between the features means that we would not see, for example, an ellipse.

instance, as a consequence of the choice of models and of the distribution of presence and absences in the environmental space, the decision boundary is not linear.

It is interesting to compare Figure 2.3 with, for example, the distribution of the raw data presented in Figure 2.1. Although we initially observed that temperature was giving us the best chance to separate the two classes, the shape of the decision boundary suggests that our classifier is considering that Corsican nuthatches enjoy colder climates with more rainfall.

2.4.4. Visualizing the trained model

We can now go through all of the pixels in the island of Corsica, and apply the model to predict the presence of *Sitta whiteheadi*. This result is reported in Figure 2.4. Because we have used training data for which we know the labels, we can also map the *outcome* of applying the model, which is to say: where are the false/true negative/positive predictions. The model seems to be making a series of false positive predictions in the northernmost part of Corsica, which may suggest that we are missing predictors relevant to this area that would refine the prediction of the suitability of the habitat.

2.4.5. What is an acceptable model?

When comparing the prediction to the spatial distribution of occurrences (Figure 2.4), it appears that the model identifies an area in the northeast where the species is likely to be present, despite limited observations. This might result in more false positives, but this is the *purpose* of running this model – if the point data were to provide us with a full knowledge of the range, there would be no point in running the model. For this reason, it is very important to nuance our interpretation of what a false-positive prediction really is. We will get back to this discussion in the next chapters, when adding more complexity to the model. For now, we have established a basic training routine for our model, and have started thinking spatially about *where* it is making errors (in space).

Note that by visualizing the type of mis-classification from our training set, we gain a better understanding of *how* the model is wrong. False positives, for example, tends to be clustered either at the western margin of the main patch of the predicted range, and in a small number of clusters in the North. False negatives are also fairly

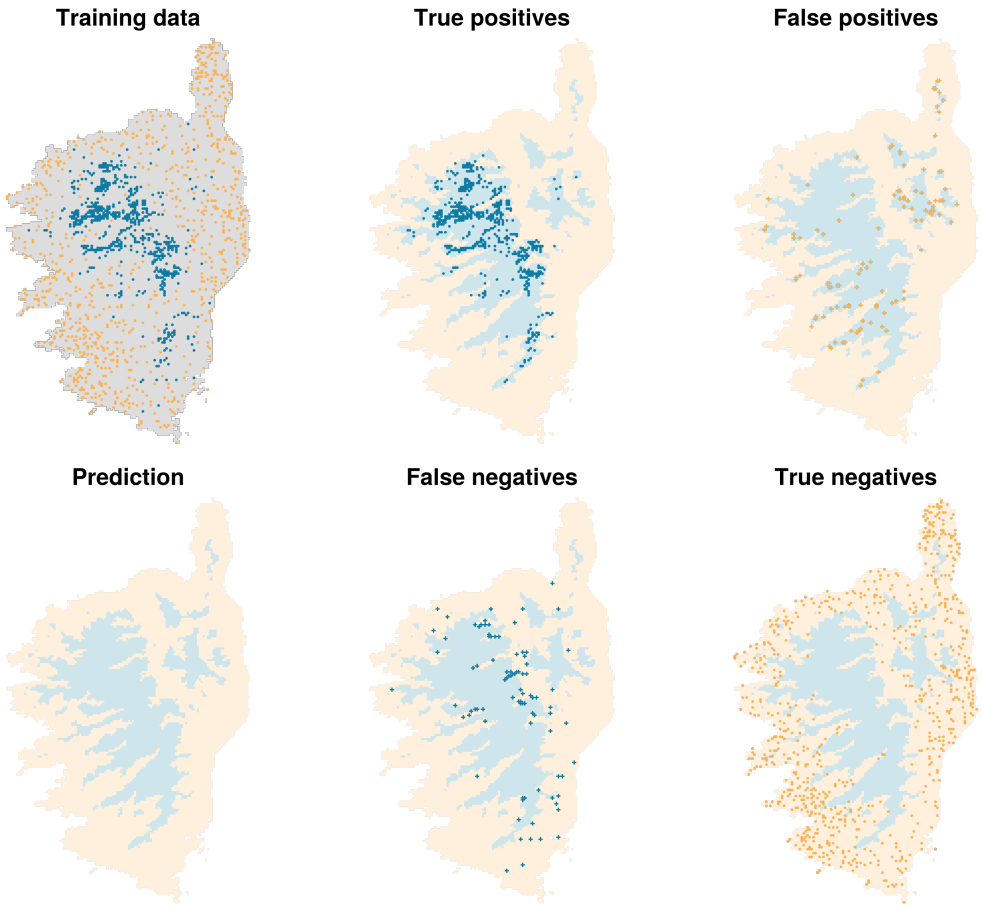


Figure 2.4.: Occurence data (left; presences are in orange and pseudo-absences in black), prediction of presences in space under the two-variables model (middle), with the four blocks of the confusion matrix also mapped. As we could have anticipated from the high values of the MCC, even this simple model does an adequate job at predicting the presence of *Sitta whiteheadi*, but would definitely stand to be improved, possibly by accounting for more features.

close to the edge of the predicted range, but clustered towards the center of the island. This is a good sign! The errors that our model is making appear to be mostly at the margin of the habitat of the species, which we know (ecologically) is more difficult to properly map out.

2.5. Conclusion

In this chapter, we introduced the Naive Bayes Classifier as a model for classification, and applied it to a data of species occurrence, in which we predicted the potential presence of the species using temperature and precipitation. Through cross-validation, we confirmed that this model gave a good enough performance (Figure 2.2), looked at the decisions that were being made by the trained model (Figure 2.3), and finally mapped the prediction and their associated errors in space (Figure 2.4). Based on this information, we concluded that the model was a reasonable first approximation of where *Sitta whiteheadi* can be present. In the next chapter, we will improve upon this model by looking at techniques to select and transform variables.

References

Abbood, A., Ullrich, A., Busche, R. & Ghozzi, S. (2020). EventEpi—A natural language processing framework for event-based surveillance. *PLOS Computational Biology*, 16, e1008277.

Barbet-Massin, M. & Jiguet, F. (2011). Back from a Predicted Climatic Extinction of an Island Endemic: A Future for the Corsican Nuthatch. *PLoS ONE*, 6, e18228.

Barbet-Massin, M., Jiguet, F., Albert, C.H. & Thuiller, W. (2012). Selecting pseudo-absences for species distribution models: how, where and how many? *Methods in Ecology and Evolution*, 3, 327–338.

Beery, S., Cole, E., Parker, J., Perona, P. & Winner, K. (2021). Species distribution modeling for machine learning practitioners: A review. *ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS)*.

Berteaux, D. (2014). *Changements climatiques et biodiversité du québec*. Presses de l'Université du Québec.

Chicco, D. & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21.

Chollet Ramampiandra, E., Scheidegger, A., Wydler, J. & Schuwirth, N. (2023). A comparison of machine learning and statistical species distribution models: Quantifying overfitting supports model interpretation. *Ecological Modelling*, 481, 110353.

Clapham, A.R., Raunkiaer, C., Gilbert-Carter, H., Tansley, A.G. & Fausboll. (1935). The life forms of plants and statistical plant geography. *The Journal of Ecology*, 23, 247.

Elith, J. & Leathwick, J.R. (2009). Species Distribution Models: Ecological Explanation and Prediction Across Space and Time. *Annual Review of Ecology, Evolution, and Systematics*, 40, 677–697.

Fick, S.E. & Hijmans, R.J. (2017). WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, 37, 4302–4315.

Guillera-Arroita, G., Lahoz-Monfort, J.J., Elith, J., Gordon, A., Kujala, H., Lentini, P.E., *et al.* (2015). Is my species distribution model fit for purpose? Matching data and models to applications. *Global Ecology and Biogeography*, 24, 276–292.

Hanberry, B.B., He, H.S. & Palik, B.J. (2012). Pseudoabsence Generation Strategies for Species Distribution Models. *PLoS ONE*, 7, e44486.

Hand, D.J. & Yu, K. (2001). Idiot's bayes: Not so stupid after all? *International Statistical Review / Revue Internationale de Statistique*, 69, 385.

Jurman, G., Riccadonna, S. & Furlanello, C. (2012). A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. *PLoS ONE*, 7, e41882.

Karger, D.N., Conrad, O., Böhner, J., Kawohl, T., Kreft, H., Soria-Auza, R.W., *et al.* (2017). Climatologies at high resolution for the earth's land surface areas. *Scientific Data*, 4.

Kupervasser, O. (2014). The mysterious optimality of Naive Bayes: Estimation of the probability in the system of "classifiers". *Pattern Recognition and Image Analysis*, 24, 1–10.

Perl, R.G.B., Avidor, E., Roll, U., Malka, Y., Geffen, E. & Gafny, S. (2022). Using eDNA presence/non-detection data to characterize the abiotic and biotic habitat requirements of a rare, elusive amphibian. *Environmental DNA*, 4, 642–653.

Powers, D.M.W. (2020). Evaluation: From precision, recall and f-measure to ROC, informedness, markedness and correlation. *arXiv*.

Whittaker, R.H. (1962). Classification of natural communities. *The Botanical Review*, 28, 1–239.

3. Preparing features

In Chapter 2, we introduced a simple classifier trained on a dataset of presence and pseudo-absences of a species (*Sitta whiteheadi*), which we predicted using the mean annual temperature as well as the annual total precipitation. This choice of variables was motivated by our knowledge of the fact that most species tend to have some temperature and precipitation they are best suited to. But we can approach the exercise of selecting predictive variables in a far more formal way, and this will form the core of this chapter. Specifically, we will examine two related techniques: variable selection, and feature engineering.

There are two reasons to think about variable selection and feature engineering – first, the variables we have may not all be predictive for the specific problem we are trying to solve; second, the variables may not be expressed in the correct "way" to solve our problem. This calls for a joint approach of selecting and transforming features. Before we do anything to our features (transformation or selection), we will discuss the important problem of data leakage, and use this discussion to establish a framework to pick hyper-parameters of the model.

3.1. The problem: optimal set of BioClim variables for the Corsican nuthatch

The BioClim suite of environmental variables are 19 measurements derived from monthly recordings of temperature and precipitation. They are widely used in species distribution modeling, despite some spatial discontinuities due to the methodology of their reconstruction (Booth 2022); this is particularly true when working from the WorldClim version (Fick & Hijmans 2017), and not as problematic when using other data products like CHELSA (Karger *et al.* 2017).

The definitions of the 19 BioClim variables are given in Table 3.1. As we can see from this table, a number of variables are either derived from the same months, or calculated through direct (sometimes linear) combinations of one another. For this reason, and because there are 19 variables, this is a good dataset to evaluate the use of variable selection and transformation.

Table 3.1.: List of the 19 BioClim variables, including indications of their calculation. The model we used in Chapter 2 used BIO1 and BIO12.

Layer	Description	Details
BIO1	Annual Mean Temperature	
BIO2	Mean Diurnal Range	Mean of monthly (max temp - min temp)
BIO3	Isothermality	BIO2/BIO7) (×100
BIO4	Temperature Seasonality	standard deviation ×100
BIO5	Max Temperature of Warmest Month	
BIO6	Min Temperature of Coldest Month	
BIO7	Temperature Annual Range	BIO5-BIO6
BIO8	Mean Temperature of Wettest Quarter	
BIO9	Mean Temperature of Driest Quarter	
BIO10	Mean Temperature of Warmest Quarter	
BIO11	Mean Temperature of Coldest Quarter	
BIO12	Annual Precipitation	
BIO13	Precipitation of Wettest Month	
BIO14	Precipitation of Driest Month	
BIO15	Precipitation Seasonality	Coefficient of Variation
BIO16	Precipitation of Wettest Quarter	
BIO17	Precipitation of Driest Quarter	
BIO18	Precipitation of Warmest Quarter	
BIO19	Precipitation of Coldest Quarter	

In this chapter, we will try to improve the model introduced in Chapter 2, by evaluating different methods to prepare our predictor variables. At this point, it is important to shift the way we think about the model: it is

This is true even when we are not transforming the variables! The identity function $f(x) = x$ is a transformation step that we can "train" (pretty easily, it turns out, as it has no parameters).

not *just* the classifier that turns the features into the prediction, but it is in fact **the entire process of transforming raw data into predictions**. This includes the transformation of these raw data, and as we will illustrate throughout this chapter, this must come with changes in the way we think about what "training" means.

3.2. What is data leakage?

Data leakage is a concept that is, if you can believe it, grosser than it sounds.

The purpose of this section is to put the fear of data leakage in you, because it can, and most assuredly *will*, lead to bad models. This is to say, as we discussed in **?@sec-gradientdescent-trainedmodel**, models that do not adequately represent the underlying data or the relationships that exists within it, in part because we have built-in some biases into them. In turn, this can eventually lead to decreased explainability of the models, which erodes trust in their predictions (Amarasinghe *et al.* 2023). As illustrated by Stock *et al.* (2023), a large number of ecological applications of machine learning are particularly susceptible to data leakage, meaning that this should be a core point of concern for us.

3.2.1. Consequences of data leakage

We take data leakage so seriously because it is one of the top ten mistakes in applied machine learning (Nisbet *et al.* 2018). Data leakage happens information "leaks" from the training conditions to the evaluation conditions; in other words, when the model is evaluated after mistakenly being fed information that would not be available in real-life situations. Note that this definition of leakage is different from another notion, namely the loss of data availability over time (Peterson *et al.* 2018).

It is worth stopping for a moment to consider what these "real-life situations" are, and how they differ from the training of the model. Most of this difference can be summarized by the fact that when we are *applying* a model, we can start from the model *only*. Which is to say, the data that have been used for the training and validation of the model may have been lost, without changing the applicability of the model: it works on entirely new data, and on new data only. The legacy of the training and validation data is only found in the parameters of the trained model. We have discussed this situation in **?@sec-crossvalidation-testing**: the test of a model is

conducted on data that have never been used for training, because we want to evaluate its performance in the conditions where it will be applied.

Because this is the behavior we want to simulate with a validation dataset, it is very important to fully disconnect the testing data from the rest of the data. We can illustrate this with an example. Let's say we want to work on a time series of population size, such as provided by the *BioTIME* project (Dornelas *et al.* 2018). One naïve approach would be to split this the time series at random into three datasets. We can use one to train the models, one to validate these models, and a last one for testing.

Congratulations! We have created data leakage! Because we are splitting our time series at random, the model will likely have been trained using data that date from *after* the start of the validation dataset. In other words: our model can peek into the future. This is highly unlikely to happen in practice, due to the ways the laws of physics work. A strategy that would prevent leakage would have been to pick a cut-off date to define the validation dataset, and then to decide how to deal with the training and testing sets.

3.2.2. Avoiding data leakage

The most common advice given in order to prevent data leakage is the "learn/predict separation" (Kaufman *et al.* 2011). Simply put, this means that whatever happens to the data used for training cannot be *simultaneously* applied to the data used for testing (or validation).

A counter-example where performing the transformation *before* the analysis is when the transformation is explicitly sought out as an embedding, where we want to predict the position of instances in the embedded space, as in *.e.g.* Runghen *et al.* (2022).

Assume that we want to transform our data using a Principal Component Analysis (PCA; Pearson (1901)). Ecologists often think of PCA as a technique to explore data (Legendre & Legendre 2012), but it is so much more than that! PCA is a model, because we can learn, from the data, a series of weights (in the transformation matrix), which we can then apply to other datasets in order to project them in the space of the projection learned from the training data.

If we have a dataset $\mathbf{X}$, which we split into two components $\mathbf{X}_0$ for training ,and $\mathbf{X}_1$ for validation, there are two ways to use a PCA to transform these data. The first is $\mathbf{T} = \mathbf{XW}$, which uses the full dataset. When we predict

the position of the validation data, we could use the transformation $\mathbf{T}_1 = \mathbf{X}_1\mathbf{W}$, but this would introduce data leakage: we have trained the transformation we apply to $\mathbf{X}_1$ using data that is already in $\mathbf{X}_1$, and therefore we have not respected the learn/predict separation.

There is a biological argument against using all the data to learn transformations anyways. Assume that we are working on a specific area in space, but want to project our bioclimatic variables using a PCA before we do this. If we work in the high arctic, is the relationship between temperature and precipitation in the Amazonian rain forest, the Serengeti desert, and the Voses mountain relevant to our problem? Surely not! For this reason, it makes sense to limit the data we use to train the transformation to the data we would have used to train the model.

There are a *lot* of peer-reviewed articles that introduce data leakage by applying PCA on bioclimatic data layers. Remember, common practices and good practices are not the same thing!

The second (correct) way to handle this situation is to perform our PCA using $\mathbf{T}_0 = \mathbf{X}_0\mathbf{W}_0$, which is to say, the weights of our PCA are derived *only* from the training data. In this situation, whenever we project the data in the validation set using $\mathbf{T}_1 = \mathbf{X}_1\mathbf{W}_0$, we respect the learn/predict separation: the transformation of $\mathbf{X}_1$ is entirely independent from the data contained in $\mathbf{X}_1$.

3.2.3. How to work in practice?

Although avoiding data leakage is a tricky problem, there is a very specific mindset we can adopt that goes a long way towards not introducing it in our analyses, and it is as follows: *every data transformation step is a modeling step that is part of the learning process*. We do not, for example, apply a PCA and train the model on the projected variables – we feed raw data into a model, the first step of which is to perform this PCA for us.

This approach works because **everything that can be represented as numbers is a model that can be trained**.

If you want to transform a variable using the z-score, this is a model! It has two parameters that you can learn from the data, μ (the average of the variable) and σ (its standard deviation). You can apply it to a data point y with $\hat{y} = (y - \mu)\sigma^{-1}$. Because this is a model, we need a dataset to learn these parameters from, and because we want to maintain the learn/predict separation, we will use the training dataset to get the values of μ_0 and σ_0. This way, when we want to get the z-score of a new observation, for example from the testing dataset, we can get it using $\hat{y}_1 = (y_1 - \mu_0)\sigma_0^{-1}$. The data transformation is entirely coming from information that was part of the training set.

One way to get the learn/predict transformation stupendously wrong is to transform our validation, testing, or prediction data using $\hat{y}_1 = (y_1 - \mu_1)\sigma_1^{-1}$. This can be easily understood with an example. Assume that the variable y_0 is the temperature in our training dataset. We are interested in making a prediction in a world that is 2 degrees hotter, uniformly, which is to say that for whatever value of y_0, the corresponding data point we use for prediction is $y_1 = y_0 + 2$. If we take the z-score of this new value based on its own average and standard deviation, a temperature two degrees warmer in the prediction data will have the same z-score as its original value, or in other words, we have hidden the fact that there is a change in our predictors! Because we learn the correct prediction from the training data, we can only apply this prediction with the parameters derived from the training data.

Treating the data preparation step as a part of the learning process, which is to say that we learn every transformation on the training set, and retain this transformation as part of the prediction process, we are protecting ourselves against both data leakage *and* the hiding of relevant changes in our predictors.

3.3. Variable selection

3.3.1. The curse of dimensionality

The number of variables we use for prediction is the number of dimensions of a problem. It would be tempting to say that adding dimensions should improve our chances to find a feature alongside which the classes become linearly separable.

Alas.

The "curse of dimensionality" is the common term of everything breaking down when the dimensions of a problem increase. In our perspective, where we rely on the resemblance between features to make a prediction, increasing the dimensions of a problem means adding features, and it has important consequences on the distance between observations. Picture two points positioned at random on the unit interval: the average distance between them is $1/3$. If we add one dimension, keeping two points but turning this line into a cube, the average distance would be about $1/2$. For a cube, about $2/3$. For n dimensions, we can figure out that the average distance grows like $\sqrt{n/6 + c}$, which is to say that when we add more dimensions, we make the

average distance between two points go to infinity. This effect is also affecting ecological studies (*e.g.* Smith *et al.* 2017).

Therefore, we need to approach the problem of "which variables to use" with a specific mindset: we want a lot of information for our model, but not so much that the space in which the predictors exist turns immense. There are techniques for this.

3.3.2. Step-wise approaches to variable selection

In order to try and decrease the dimensionality of a problem, we can attempt to come up with a method to decide which variables to include, or to remove, from a model. This practice is usually called "stepwise" selection, and is the topic of *intense* debate in ecology, although several studies point to the fact that there is rarely a best technique to select variables (Murtaugh 2009), that the same data can usually be adequately described by competing models (WHITTINGHAM *et al.* 2006), and that classifiers can show high robustness to the inclusion of non-informative variables (Fox *et al.* 2017). Situations in which variable selection has been shown top be useful is the case of model transfer (Petitpierre *et al.* 2016), or (when informed by ecological knowledge), the demonstration that classes of variables had no measurable impact on model performance (Thuiller *et al.* 2004).

Why, so, should we select the variables we put in our models?

The answer is simple: we seek to solve a specific problem in an optimal way, where "optimal" refers to the maximization of a performance measure we decided upon *a priori*. In our case, this is the MCC. Therefore, an ideal set of predictors is the one that, given our cross-validation strategy, maximizes our measure of performance.

3.3.2.1. Forward selection

In forward selection, assuming that we have f features, we start by building f models, each using one feature. For example, using the BioClim variables, m_1 would be attempting to predict presences and absences based only on temperature. Out of these models, we retain the variable given by $\text{argmax}_f \text{MCC}(m_f)$, where $\text{MCC}(m_f)$ is the average value of MCC for the f-th model on the validation datasets. This is the first variable we add to

our set of selected variables. We then train $f - 1$ models, and then again add the variable that leads to the best possible *increase* in the average value of the MCC. When we cannot find a remaining variable that would increase the performance of the model, we stop the process, and return the optimal set of variables. Forward selection can be constrained by, instead of starting from variables one by one, starting from a pre-selected set of variables that will always be included in the model.

There are two important things to consider here. First, the set of variables is only optimal under the assumptions of the stepwise selection process: the first variable is the one that boosts the predictive value of the model the most *on its own*, and the next variables *in the context of already selected variables*. Second, the variables are evaluated on the basis of their ability to *improve the performance of the model*; this does not imply that they are relevant to the ecological processes happening in the dataset. Infering mechanisms on the basis of variable selection is foolish (Tredennick *et al.* 2021).

3.3.2.2. Backward selection

The opposite of forward selection is backward selection, in which we start from a complete set of variables, then remove the one with the *worst* impact on model performance, and keep proceeding until we cannot remove a variable without making the model worse. The set of variables that remain will be the optimal set of variables. In almost no cases will forward and backward selection agree on which set of variables is the best – we have to settle this debate by either picking the model with the least parameters (the most parsimonious), or the one with the best performance.

Why not evaluate all the combination of variables?

Keep in mind that we do not know the number of variables we should use; therefore, for the 19 BioClim variables, we would have to evaluate $\sum_f \binom{19}{f}$, which turns out to be an *immense* quantity (for example, $\binom{19}{9} = 92378$). For this reason, a complete enumeration of all variable combinations would be extremely wasteful.

3.3.3. Removal of colinear variables

Co-linearity of variables is challenging for all types of ecological models (Graham 2003). In the case of species distribution models (De Marco & Nóbrega 2018), the variables are expected to be strongly auto-correlated, both because they have innate spatial auto-correlation, and because they are derived from a smaller set of raw data (Dormann *et al.* 2012). For this reason, it is a good idea to limit the number of colinear variables.

THIS PARAGRAPH IS NOT FINISHED

3.4. Multivariate transformations

3.4.1. PCA-based transforms

Principal Component Analysis (PCA) is one of the most widely used multi-variate techniques in ecology, and is a very common technique to prepare variables in applied machine learning. One advantage of PCA is that it serves both as a way to remove colinearity, in that the principal components are orthogonal, and as a way to reduce the dimensionality of the problem as long as we decide on a threshold on the proportion of variance explained, and only retain the number of principal components needed to reach this threshold. For applications where the features are high-dimensional, PCA is a well established method to reduce dimensionality *and* extract more information in the selected principal components (Howley *et al.* 2005). In PCA, the projection matrix $\mathbf{P}$ is applied to the data using $\mathbf{P}^\top(\mathbf{x} - \square)$, where $\mathbf{x}$ is the feature matrix with means $\square$. Typically, the dimensions of $\mathbf{P}$ are *lower* than the dimensions of $\mathbf{x}$, resulting in fewer dimensions to the problem. Cutoffs on the dimensions of $\mathbf{P}$ are typically expressed as a proportion of the overall variance maintained after the projection. Variants of PCA include kernel PCA (Schölkopf *et al.* 1998), using a higher-dimensional space to improve the separability of classes, and probabilistic PCA (Tipping & Bishop 1999), which relies on modeling the data within a latent space with lower dimensionality.

3.4.2. Whitening transforms

Another class of potentially very useful data transformations is whitening transforms, which belongs to the larger category of decorrelation methods. These methods do not perform any dimensionality reduction, but instead remove the covariance in the datasets. Whitening has proven to be particularly effective at improving the predictive ability of models applied to data with strong covariance structure (Koivunen & Kostinski 1999). In essence, given a matrix of features $\mathbf{x}$, with averages $\square$ and covariance $\mathbf{C}$, a whitening transform $\mathbf{W}$ is *one of the matrices* that satisfies $\mathbf{W}^\top \mathbf{C} \mathbf{W} = \mathbf{I}$.

In other words, the whitening transform results in a new set of features with unit variance and no covariance: the dimensionality of the problem remains the same but the new random variables are independent. Given any dataset with covariance matrix $\mathbf{C}$, if any $\mathbf{W}$ is a whitening transform, then so to are any matrices $\mathbf{WR}$ where $\mathbf{R}$ performs a rotation with $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$. The optimal whitening transform can be derived through a variety of ways (see *e.g.* Kessy *et al.* 2018). The whitening transform is applied to the input vector using $\mathbf{W}^\top(\mathbf{x} - \square)$: this results in new random variables that have a mean of 0, and unit variance. The new input vector after the transformation is therefore an instance of "white noise" (Vasseur & Yodzis 2004).

When performing Whitening transformation, the first variable in the new space will usually have pretty high correlation to the first raw variable. But because the purpose of Whitening is to remove the covariance (and ensure unit variance), this gets less and less true as we increase the rank of the variables. By the time the last selected variable is reached, we should expect it to be almost purely noise.

3.5. Application: optimal variables for Corsican nuthatch

Before we start, we can re-establish the baseline performance of the model from Chapter 2. In this (and the next) chapters, we will perform k-folds cross-validation (see **?@sec-crossvalidation-kfolds** for a refresher), using $k = 15$. This strategy gives an average MCC of 0.727, which represents our "target": any model with a higher MCC will be "better" according to our criteria.

In a sense, this initial model was *already* coming from a variable selection process, only we did not use a quantitative criteria to include variables. And so, it is a good idea to evaluate how our model performed, relative to

a model including *all* the variables. Running the NBC again using all 19 BioClim variables from Table 3.1, we get an average MCC on the validation data of 0.748. This is a small increase, but an increase nevertheless – our dataset had information that was not captured by temperature and precipitation. But this model with all the variables most likely includes extraneous information that does not help, or even hinders, the predictive ability of our model. Therefore, there is probably a better version of the model somewhere, that uses the optimal set of variables, potentially with the best possible transformation applied to them.

In this section, we will start by evaluating the efficiency of different approaches to variable selection, then merge selection and transformation together to provide a model that is optimal with regards to the training data we have (the workflow is outlined in **?@fig-predictors-workflow**). In order to evaluate the model, we will maintain the use of the MCC; in addition, we will report the PPV and NPV (like in Chapter 2), as well as the accuracy and True-Skill Statistic (TSS). The TSS is defined as the sum of true positive and true negative rates, minus one, and is an alternative measure to the MCC (although it is more sensitive to some biases). Although several authors have advocated for the use of TSS (ALLOUCHE *et al.* 2006), Leroy *et al.* (2018) have an interesting discussion of how the TSS is particularly sensitive to issues in the quality of (pseudo) absence data. For this reason, and based on the literature we covered in Chapter 2, there is no strong argument against using MCC as our selection measure.

To prevent the risk of interpreting the list of variables that have been retained by the model, we will *not* make a list of which they are (yet). This is because, in order to discuss the relative importance of variables, we need to introduce a few more concepts and techniques, which will not happen until **?@sec-explanations**; at this point, we will revisit the list of variables identified during this chapter, and compare their impact on model performance to their actual importance in explaining predictions.

In Chapter 4, we will revisit the question of how the MCC is "better", and spend more time evaluating alternatives. For now, we can safely *assume* that MCC is the best.

3.5.1. Variable selection

We will perform four different versions of stepwise variable selection. Forward, forward from a pre-selected set of two variables (temperature and precipitation), backward, and based on the Variance Inflation Factor (with a cutoff of 10). The results are presented in Table 3.2.

Table 3.2.: Consequences of different variable selection approaches on the performance of the model, as evaluated by the MCC averaged over the validation datasets. The highest MCC value, corresponding to the best model, is in bold.

Model	**Variables**	**MCC**
Chapter 2 baseline	1,12	0.727
All variables		0.748
Fwd. sel.	8,4,7,6,15	**0.777**
Constr. sel.	1,12,10,3,8	0.773
Backw. sel.	1,2,3,5,6,7,8,9,10,11,14,15,17	0.775
Var. infl. fac.	2,6,18	0.754

The best model is given by forward selection, although backwards selection also gives a very close performance. At this point, we may decide to keep these two strategies, and evaluate the effect of different transformations of the data.

3.5.2. Variable transformation

Based on the results from Table 3.2, we retain forward and backwards selection as our two stepwise selection methods, and now apply an additional transformation (as in **?@fig-predictors-workflow**) to the subset of the variables. The results are presented in Table 3.3. Based on these results, and using the MCC as the criteria for the "best" model, we see that combining forward selection with a whitening transform gives the best predictive performance. Note that the application of a transformation *does* change the result of variable selection, as evidences by the fact that the number of retained variables changes when we apply a transformation.

Table 3.3.: Model performance when coupling variable selection with variable transformation.

Variable selection	Transformation	Nb. variables	Latent var.	Var. expl.	MCC (val.)	MCC (train)
Chapter 2 baseline		2			0.727	0.729
	PCA	2	1	1.00	0.476	0.478
Fwd. sel.		5			0.777	0.780
	PCA	4	4	1.00	**0.796**	0.797
Backw. sel.		13			0.775	0.777
	PCA	15	3	0.99	0.773	0.773
Constr. sel.		5			0.773	0.774
	PCA	6	2	1.00	0.484	0.485

3.5.3. Variable importance

Before moving on, it is worth taking a moment to think about what we have done so far. We have optimized two components of our model: the transformation of the raw data before they are fed to the classifier (we use a PCA for this step), and then (or simultaneously) the list of variables which we put into the model. In Chapter 2, we had assumed that temperature and precipitation were important variables based on decades of ecological folklore (or, as we say, "expert knowledge"). But what about the list of variables currently retained in this model? Although it may be tempting to come up with *ad hoc* explanations for why they matter, we can rather ask the question of "how much do these variables actually matter?".

The process we will adopt is simple: we train a model on a training set $\mathbf{X}_t$, and measure its performance on a validation set $\mathbf{X}_v$. The performance of the model on the validation we note p_v. Because we want to avoid re-training our model (because we may not have access to the training data, in practice), the only thing we can act on is the validation set. The process we will use is to identify a feature j, and then across the entire validation set, shuffle its j-th column. At this point, we can re-apply our model to this shuffled validation set (let's note it $\mathbf{X}_v^{(j)}$), and measure the performance of the model on this perturbed dataset as $p_v^{(j)}$. The impact of shuffling the variable j is given as $m_j = \|p_v - p_v^{(j)}\|$.

We can repeat this process a number of times (because we are relying on a shuffle, so it is better to perform this process many times over), and report the importance of variable j as $\bar{m}_j$. This importance is expressed on the scale of the measure of performance we use. When it is large, it means that shuffling the variable j has a very large impact on the performance of the model: it is important to know the correct value of this feature to make the "right" prediction. Because the absolute score $\bar{m}_j$ is not very informative, we can instead report a *relative* importance of a variable, which is simply $\bar{m}_j / \sum_{i \in \text{var.}} \bar{m}_i$. These will sum to one, and can interpreted as the proportional importance of the selected variables.

But what does the process of shuffling simulate? Low quality data, sampling, wrong information, loss of information etc

3.5.4. Model selection and output

In Table 3.2 and Table 3.3, we have evaluated a series of several modeling strategies, defined by a variable selection and transformation technique. Using the MCC as our reference for what constitutes the best model, we can now apply the model to the relevant set of predictors, in order to see how these refinements result in a new predicted range for the species.

In Figure 3.2, we see how the use of a PCA turns co-linear variables into a much more manageable set of points. Although we can still think of our model in terms of the untransformed predictors we feed it, the classifier will only attempt to learn parameters from the projected data. These results are presented in Figure 3.3.

3.6. The Rashomon effect

The Rashomon effect is defined as the fact that, given a single problem, we can find many models that will achieve, on average, the same performance (or the same loss). In other words, the same event (the prediction we want to make) can often admit several seemingly reasonable yet incompatible interpretations (just like in the movie usually invoked as an example of the Rashomon effect: *Hoodwinked!*). So far, we have assumed that we were able to pick the *best* model, but the *best* is often a relatively minor improvement over the second-best model.

This chapter is a good illustration of the Rashomon effect. Most models in Table 3.3 have a very similar performance on the same problem. We know this because we evaluated the performance of these models on the *exact* same folds, training, and validation points. By picking the best model, we had to discard a large number of models that were, all things considered, similarly good.

We will talk about the Rashomon effect again in Chapter 4, in **?@sec-bagging**, and further in **?@sec-counterfactuals**.

This is not really a problem if all we care about is "making a good prediction". Even then, a similar loss or MCC score can be achieved through different predictions once mapped out. But where the Rashomon effect is particularly problematic is when we start attempting to provide *explanations* for the predictions. All the best models (under different variable selection and data transformation schemes) had selected different features. What this means is that, if we attempt to explain a specific prediction with regards to, for example, annual precipitation, this task is only meaningful in a model that uses annual precipitation as a feature.

The Rashomon effect manifests at all scales of the machine learning process, and we will see instances of it again in the following chapters.

3.7. Conclusion

In this chapter, we have discussed the issues with dimensionality and data leakage, and established a methodology to reduce the number of dimensions (and possible re-project the variables) while maintaining the train/predict separation. This resulted in a model whose performance (as evaluated using the MCC) increased quite significantly, which resulted in the predicted range of *Sitta whiteheadi* changing in space.

In Chapter 4, we will finish to refine this model, by considering that the NBC is a probabilistic classifier, and tuning various hyper-parameters of the model using learning curves and thresholding. This will result in the final trained model, the behavior of which we will explore in **?@sec-explanations**, to understand *how* the model makes predictions.

References

ALLOUCHE, O., TSOAR, A. & KADMON, R. (2006). Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). *Journal of Applied Ecology*, 43, 1223–1232.

Amarasinghe, K., Rodolfa, K.T., Lamba, H. & Ghani, R. (2023). Explainable machine learning for public policy: Use cases, gaps, and research directions. *Data & Policy*, 5.

Booth, T.H. (2022). Checking bioclimatic variables that combine temperature and precipitation data before their use in species distribution models. *Austral Ecology*, 47, 1506–1514.

De Marco, P. & Nóbrega, C.C. (2018). Evaluating collinearity effects on species distribution models: An approach based on virtual species simulation. *PLOS ONE*, 13, e0202403.

Dormann, C.F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., *et al.* (2012). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*, 36, 27–46.

Dornelas, M., Antão, L.H., Moyes, F., Bates, A.E., Magurran, A.E., Adam, D., *et al.* (2018). BioTIME: A database of biodiversity time series for the Anthropocene. *Global Ecology and Biogeography*, 27, 760–786.

Fick, S.E. & Hijmans, R.J. (2017). WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, 37, 4302–4315.

Fox, E.W., Hill, R.A., Leibowitz, S.G., Olsen, A.R., Thornbrugh, D.J. & Weber, M.H. (2017). Assessing the accuracy and stability of variable selection methods for random forest modeling in ecology. *Environmental Monitoring and Assessment*, 189.

Graham, M.H. (2003). CONFRONTING MULTICOLLINEARITY IN ECOLOGICAL MULTIPLE REGRESSION. *Ecology*, 84, 2809–2815.

Howley, T., Madden, M.G., O'Connell, M.-L. & Ryder, A.G. (2005). The effect of principal component analysis on machine learning accuracy with high dimensional spectral data. Springer London, pp. 209–222.

Karger, D.N., Conrad, O., Böhner, J., Kawohl, T., Kreft, H., Soria-Auza, R.W., *et al.* (2017). Climatologies at high resolution for the earth's land surface areas. *Scientific Data*, 4.

Kaufman, S., Rosset, S. & Perlich, C. (2011). Leakage in data mining. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Kessy, A., Lewin, A. & Strimmer, K. (2018). Optimal Whitening and Decorrelation. *The American Statistician*, 72, 309–314.

Koivunen, A.C. & Kostinski, A.B. (1999). The Feasibility of Data Whitening to Improve Performance of Weather Radar. *Journal of Applied Meteorology*, 38, 741–749.

Legendre, P. & Legendre, L. (2012). *Numerical ecology*. Developments in environmental modelling. Third English edition. Elsevier, Oxford, UK.

Leroy, B., Delsol, R., Hugueny, B., Meynard, C.N., Barhoumi, C., Barbet-Massin, M., *et al.* (2018). Without quality presence–absence data, discrimination metrics such as TSS can be misleading measures of model performance. *Journal of Biogeography*, 45, 1994–2002.

Murtaugh, P.A. (2009). Performance of several variable-selection methods applied to real ecological data. *Ecology Letters*, 12, 1061–1068.

Nisbet, R., Miner, G., Yale, K., Elder, J.F. & Peterson, A.F. (2018). *Handbook of statistical analysis and data mining applications*. Second edition. Academic Press, London.

Pearson, K. (1901). LIII. *On lines and planes of closest fit to systems of points in space*. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2, 559–572.

Peterson, A.T., Asase, A., Canhos, D., Souza, S. de & Wieczorek, J. (2018). Data leakage and loss in biodiversity informatics. *Biodiversity Data Journal*, 6.

Petitpierre, B., Broennimann, O., Kueffer, C., Daehler, C. & Guisan, A. (2016). Selecting predictors to maximize the transferability of species distribution models: lessons from cross-continental plant invasions. *Global Ecology and Biogeography*, 26, 275–287.

Runghen, R., Stouffer, D.B. & Dalla Riva, G.V. (2022). Exploiting node metadata to predict interactions in bipartite networks using graph embedding and neural networks. *Royal Society Open Science*, 9.

Schölkopf, B., Smola, A. & Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10, 1299–1319.

Smith, M.L., Ruffley, M., Espíndola, A., Tank, D.C., Sullivan, J. & Carstens, B.C. (2017). Demographic model selection using random forests and the site frequency spectrum. *Molecular Ecology*, 26, 4562–4573.

Stock, A., Gregr, E.J. & Chan, K.M.A. (2023). Data leakage jeopardizes ecological applications of machine learning. *Nature Ecology & Evolution*.

Thuiller, W., Araújo, M.B. & Lavorel, S. (2004). Do we need land-cover data to model species distributions in Europe? *Journal of Biogeography*, 31, 353–361.

Tipping, M.E. & Bishop, C.M. (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61, 611–622.

Tredennick, A.T., Hooker, G., Ellner, S.P. & Adler, P.B. (2021). A practical guide to selecting models for exploration, inference, and prediction in ecology. *Ecology*, 102.

Vasseur, D.A. & Yodzis, P. (2004). THE COLOR OF ENVIRONMENTAL NOISE. *Ecology*, 85, 1146–1152.

WHITTINGHAM, M.J., STEPHENS, P.A., BRADBURY, R.B. & FRECKLETON, R.P. (2006). Why do we still use stepwise modelling in ecology and behaviour? *Journal of Animal Ecology*, 75, 1182–1189.

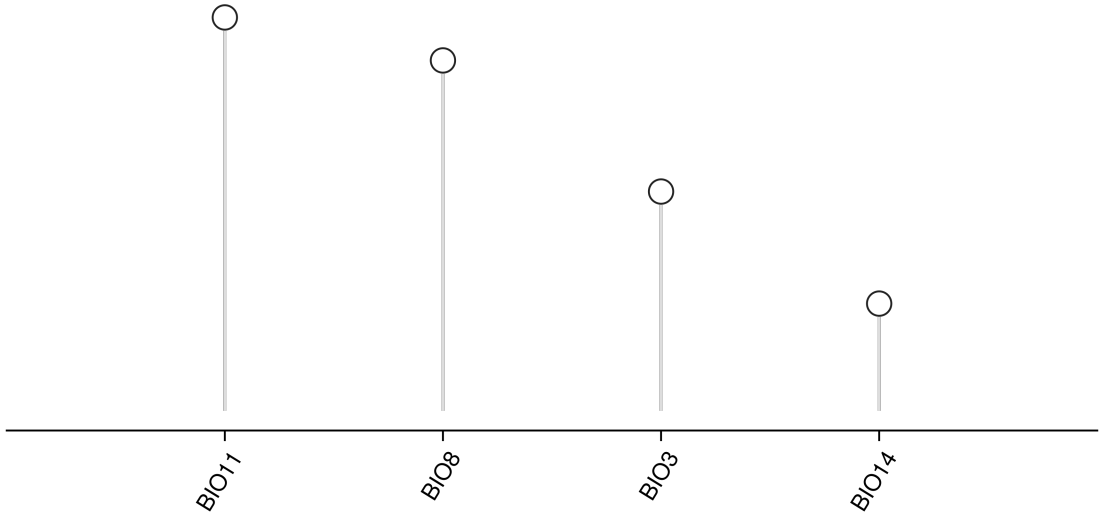


Figure 3.1.: Relative importance of the variables selected as part of the best model. The importance of the variables has been measured by comparing their performance on validation sets before and after shuffling the column of the feature matrix corresponding to the variable to test.

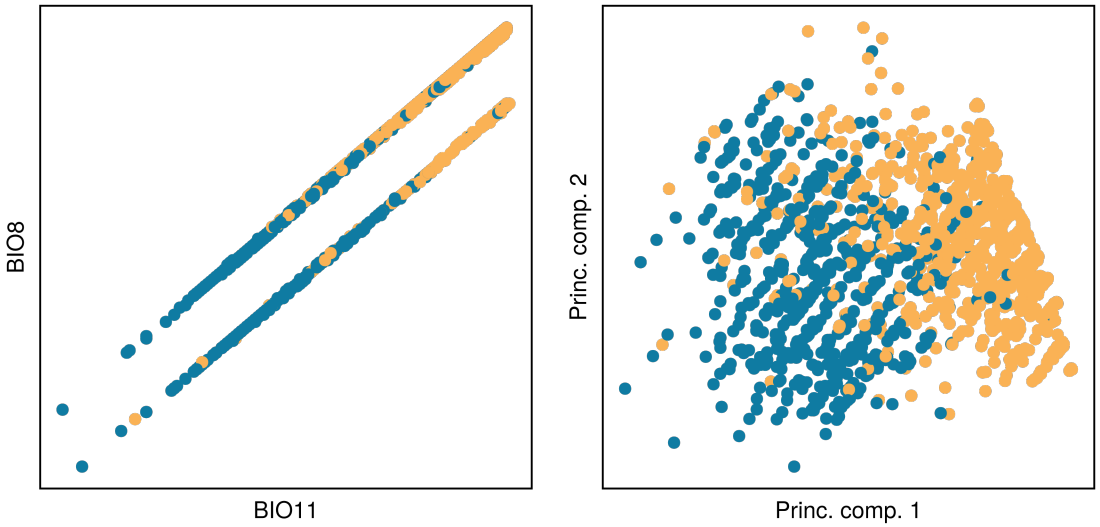


Figure 3.2.: orig var vs. their transformation

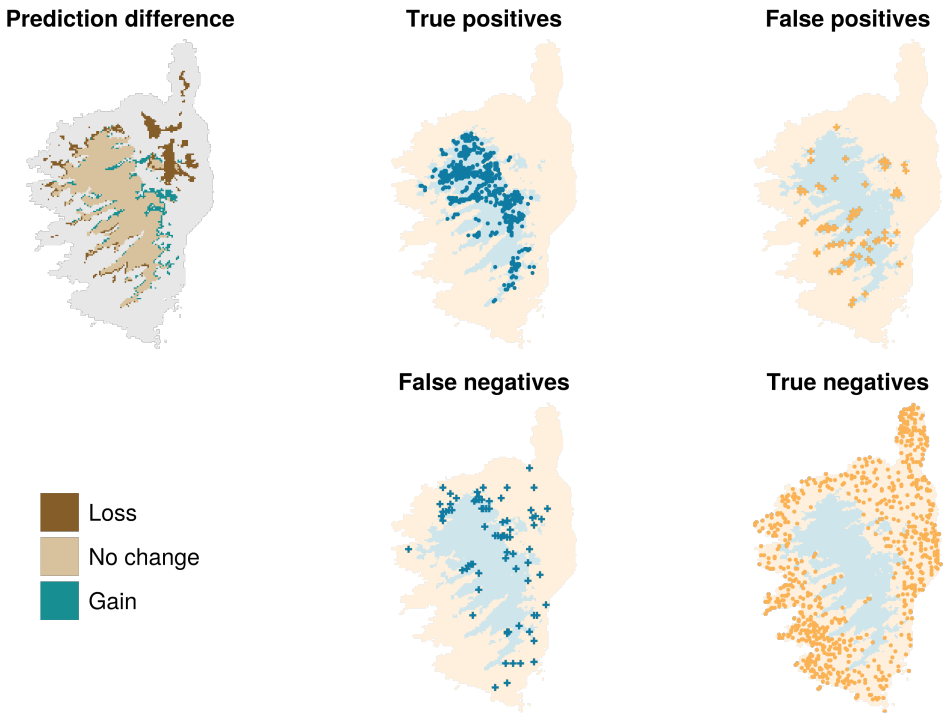


Figure 3.3.: Consequences of different variable transformations on the predicted range of Sitta whiteheadi. Note that the small area of predicted presence in the Cap Corse (the Northern tip) has disappeared with the new set of variables and their optimal transformation.

4. Tuning hyper-parameters

In **?@sec-gradientdescent**, we represented the testing and training loss of a model as a function of the number of gradient descent steps we had made. This sort of representation is very useful to figure out how well our model is learning, and is called, appropriately enough, a learning curve. We further discussed that the learning rate (and possibly the regularization rate), and the number of epochs, where *hyper*-parameters of the model. An hyper-parameter is usually defined as a parameter of the model that is *controlling* the learning process, but is not itself modified through learning (Yang & Shami 2020). Hyper-parameters usually need to be determined *before* the training starts (Claesen & De Moor 2015), but there are various strategies to optimize them. In this chapter, we will produce learning curves to find the optimal values of an hyper-parameter of the model we developed in Chapter 2 and Chapter 3 (the threshold at which we consider that a probability is high enough to be considered a positive prediction).

We will illustrate this using an approach called moving-threshold classification, and additionally discuss how we can conduct searches to tune several hyper-parameters at once. There are many techniques to sample multiple parameters at the same time, including Latin hypercube sampling (Huntington & Lyrintzis 1998), successive halvings (Jamieson & Talwalkar 2016), orthogonal sampling (McKay *et al.* 1979), and grid searches. The common point to all of these approaches are that they generate a combination of hyper-parameters, which are used to train the model, and measures of performance are then used to pick the best possible combination of hyper-parameters. In the process of doing this, we will also revisit the question of why the MCC is a good measure of the classification performance, as well as examine tools to investigate the "right" balance between false/true positive rates. At the end of this chapter, we will have produced a very good model for the distribution of the Corsican nuthatch, which we will then *explain* in **?@sec-explanations**.

4.1. Classification based on probabilities

When first introducing classification in Chapter 2 and Chapter 3, we used a model that returned a deterministic answer, which is to say, the name of a class (in our case, this class was either "present" or "absent"). But a lot of classifiers return quantitative values, that correspond to (proxies for) the probability of the different classes. Nevertheless, because we are interested in solving a classification problem, we need to end up with a confusion table, and so we need to turn a number into a class. In the context of binary classification (we model a yes/no variable), this can be done using a threshold for the probability.

Note that the quantitative value returned by the classifier does not *need* to be a probability; it simply needs to be on an interval (or ratio) scale.

The idea behind the use of thresholds is simple: if the classifier output $\hat{y}$ is larger than (or equal to) the threshold value τ, we consider that this prediction corresponds to the positive class (the event we want to detect, for example the presence of a species). In the other case, this prediction corresponds to the negative class. Note that we do not, strictly, speaking, require that the value $\hat{y}$ returned by the classifier be a probability. We can simply decide to pick τ somewhere in the support of the distribution of $\hat{y}$.

The threshold to decide on a positive event is an hyper-parameter of the model. In the NBC we built in Chapter 2, our decision rule was that $p(+) > p(-)$, which when all is said and done (but we will convince ourselves of this in Section 4.4.1), means that we used $\tau = 0.5$. But there is no reason to assume that the threshold needs to be one half. Maybe the model is overly sensitive to negatives. Maybe there is a slight issue with our training data that bias the model predictions. And for this reason, we have to look for the optimal value of τ.

There are two important values for the threshold, at which we know the behavior of our model. The first is $\tau = \min(\hat{y})$, for which the model *always* returns a negative answer; the second is, unsurprisingly, $\tau = \max(\hat{y})$, where the model *always* returns a positive answer. Thinking of this behavior in terms of the measures on the confusion matrix, as we have introduced them in Chapter 2, the smallest possible threshold gives only negatives, and the largest possible one gives only positives: they respectively maximize the false negatives and false positives rates.

4.1.1. The ROC curve

This is a behavior we can exploit, as increasing the threshold away from the minimum will lower the false negatives rate and increase the true positive rate, while decreasing the threshold away from the maximum will

lower the false positives rate and increase the true negative rate. If we cross our fingers and knock on wood, there will be a point where the false events rates have decreased as much as possible, and the true events rates have increased as much as possible, and this corresponds to the optimal value of τ for our problem.

We have just described the Receiver Operating Characteristic (ROC; Fawcett (2006)) curve! The ROC curve visualizes the false positive rate on the x axis, and the true positive rate on the y axis. The area under the curve (the ROC-AUC) is a measure of the overall performance of the classifier (Hanley & McNeil 1982); a model with ROC-AUC of 0.5 performs at random, and values moving away from 0.5 indicate better (close to 1) or worse (close to 0) performance. The ROC curve is a description of the model performance across all of the possible threshold values we investigated!

The ROC curve can be used to rule out a model that performs worse than the no-skill classifier.

4.1.2. The PR curve

One very common issue with ROC curves, is that they are overly optimistic about the performance of the model, especially when the problem we work on suffers from class imbalance, which happens when observations of the positive class are much rarer than observations of the negative class. In ecology, this is a common feature of data on species interactions (Poisot *et al.* 2023). In addition, although a good model will have a high ROC-AUC, a bad model can get a high ROC-AUC too (Halligan *et al.* 2015); this means that ROC-AUC alone is not enough to select a model.

An alternative to ROC is the PR (for precision-recall) curve, in which the positive predictive value is plotted against the true-positive rate; in other words, the PR curve (and therefore the PR-AUC) quantify whether a classifier makes reliable positive predictions, both in terms of these predictions being associated to actual positive outcomes (true-positive rate) and not associated to actual negative outcomes (positive predictive value). Because the PR curve uses the positive predictive values, it captures information that is similar to the ROC curve, but is in general more informative (Saito & Rehmsmeier 2015).

4.1.3. The TPTS curve

Becker *et al.* (2022) developed a variant of the ROC curve meant to be used when the validation data are *only* composed of the predictive class; there are a number of situations when this is a reasonable assumption. In the

original article, the testing data were reported positive detection of beta-coronaviruses in bat species, which can be seen as a positive-only event since negative tests are unlikely to be reported [the "file-drawer effect"; Pautasso (2010)]. In this context, evaluating the model by accounting for negative testing data introduces biases, as we do not have access to novel negative instances.

The idea behind the TPTS curve is to instead evaluate the sensitivity of the model as a function of the prevalence that would have been observed *during training* using a specific threshold. At higher thresholds, all instances are predicted positive, but the TPTS curve quantifies how reliably high the specificity can be while keeping the threshold as low as possible. As for the ROC curve, a good model will get a high specificity even at a low threshold.

4.2. A note on cross-entropy loss

In **?@sec-gradientdescent**, we used loss functions to measure the progress of our learning algorithm. Unsurprisingly, loss functions exist for classification tasks too. One of the most common is the cross-entropy (or log-loss), which is defined at the scale of each prediction as

$$L_i = -\left[y_i \times \log p_i + (1 - y_i) \times \log\left(1 - p_i\right)\right], \tag{4.1}$$

where y_i is the actual class, and p_i is the probability associated to the positive class for an input x_i. The loss of a model is simply the average of the contributions of individual points. Note that the log-loss is very similar to Shannon's measure of entropy, and in fact can be expressed based on the Kullback-Leibler divergence of the distributions of y and p. All of this is to say say that the cross-entropy loss measures how much information about y is conveyed by p.

In this chapter, we use measures like the MCC that describe the performance of a classifier when the predictions are done, but log-loss is useful when there are multiple epochs of training. Neural networks used for classification commonly use log-loss as a loss function; note that the gradient of the log-loss function is very easy to calculate, and that gives it its usefulness as a measure of the advancement of the learning process.

As a gentle introduction to what's next in this chapter, we can start using the loss to optimize the training process. Most things related to cross-validation *are* hyper-parameters, in that they will regulate our ability to conduct the training process under the best possible conditions. In this section, we will revisit the cross-validation of the model, by investigating how the choice of cross-validation can lead to adverse outcomes. In Figure 4.1, we show how increasing the proportion of datasets retained for model evaluations using holdout cross-validation **?@sec-crossvalidation-holdout** can change our ability to produce a good model.

Specifically, we compare the cross-entropy loss estimated from 100 replicated attempts at training the model for the training and validation data. As the proportion of holdout data increases, which is to say that we train the model with fewer and fewer data points, notice that the MCC on the training data *increases*, but the MCC on the validation data *decreases*. This is a sign of overfitting, as the model trained with too much validation data is not seeing enough examples to infer a general enough distribution of the conditions leading to a correct classification of presence/absence observations. **TODO** CHANGE THIS PARAGRAPH NOT MCC

Performing this type of analysis is crucial, as it will help us figure out the correct conditions under which a model can be trained. In this case, if we decided to use holdout cross-validation (which we do not, we will keep using k-folds!) it appears that the model performance can be reliably be estimated even with low holdout proportions. The usual cutoff of 30% of data used for training using holdout would give reliable estimates of model performance.

Remember from **?@sec-crossvalidation-kfolds** that with k-fold, k is an hyper-parameter; it can be tuned in the exact same way! The value of k used in these chapters has been picked because it gives adequate performance.

This simple example served as an illustration of what a learning curve looks like. In the rest of this chapter, we will focus on tuning an hyper-parameter from the model itself (the probability threshold for attribution of the positive class), and see how we can re-construct a general approach from thinking about the components of the confusion table.

4.3. How to optimize the threshold?

In order to understand the optimization of the threshold, we first need to understand how a model with thresholding works. When we run such a model on multiple input features, it will return a list of probabilities, for example $[0.2, 0.8, 0.1, 0.5, 1.0]$. We then compare all of these values to an initial threshold, for example $\tau = 0.05$, giving us a vector of Boolean values, in this case $[+, +, +, +, +]$. We can then compare this classified output

to a series of validation labels, *e.g.* $[-, +, -, -, +]$, and report the performance of our model. In this case, the very low thresholds means that we accept any probability as a positive case, and so our model is very strongly biased (towards false positives). We then increase the threshold, and start again.

As we have discussed in Section 4.1, moving the threshold is essentially a way to move in the space of true/false rates. As the measures of classification performance capture information that is relevant in this space, there should be a value of the threshold that maximizes one of these measures. Alas, no one agrees on which measure this should be (Perkins & Schisterman 2006; Unal 2017). The usual recommendation is to use the True Skill Statistic, also known as Youden's J (Youden 1950). The biomedical literature, which is quite naturally interested in getting the interpretation of tests right, has established that maximizing this value brings us very close to the optimal threshold for a binary classifier (Perkins & Schisterman 2005). In a simulation study, using the True Skill Statistic gave good predictive performance for models of species interactions (Poisot 2023).

Some authors have used the MCC as a measure of optimality (Zhou & Jakobsson 2013), as it is maximized *only* when a classifier gets a good score for the basic rates of the confusion matrix. Based on this information, Chicco & Jurman (2023) recommend that MCC should be used to pick the optimal threshold *regardless of the question*, and I agree with their assessment. A high MCC is always associated to a high ROC-AUC, TSS, etc., but the opposite is not necessarily true. This is because the MCC can only reach high values when the model is good at *everything*, and therefore it is not possible to trick it. In fact, previous comparisons show that MCC even outperform measures of classification loss (Jurman *et al.* 2012).

For once, and after over 15 years of methodological discussion, it appears that we have a conclusive answer! In order to pick the optimal threshold, we find the value that maximizes the MCC. Note that in previous chapters, we already used the MCC as a our criteria for the best model, and now you know why.

4.4. Application: improved Corsican nuthatch model

In this section, we will finish the training of the model for the distribution of *Sitta whiteheadi*, by picking optimal hyper-parameters, and finally reporting its performance on the testing dataset. At the end of this chapter, we will therefore have established a trained model, that we will use in **?@sec-explanations** to see how each prediction emerges.

4.4.1. Making the NBC explicitly probabilistic

In Chapter 2, we have expressed the probability that the NBC recommends a positive outcome as

$$P(+|x) = \frac{P(+)}{P(x)} P(x|+) \,,$$

and noted that because $P(x)$ is constant across all classes, we could simplify this model as $P(+|x) \propto P(+)P(x|+)$. But because we know the only two possible classes are $+$ and $-$, we can figure out the expression for $P(x)$. Because we are dealing with probabilities, we know that $P(+|x) + P(-|x) = 1$. We can therefore re-write this as

$$\frac{P(+)}{P(x)} P(x|+) + \frac{P(-)}{P(x)} P(x|-) = 1$$

which after some reorganization (and note that $P(-) = 1 - P(+)$), results in

$$P(x) = P(+)P(x|+) + P(-)P(x|-) \,.$$

This value $P(x)$ is the "evidence" in Bayesian parlance, and we can use this value explicitly to get the prediction for the probability associated to the class $+$ using the NBC.

Note that we can see that using the approximate version we used so far (the prediction is positive if $P(+)P(x|+) > P(-)P(x|-)$) is equivalent to saying that the prediction is positive whenever $P(+|x) > \tau$ with $\tau = 0.5$. In the next sections, we will challenge the assumption that 0.5 is the optimal value of τ.

In Figure 4.2, we show the effect of moving the threshold from 0 to 1 on the value of the MCC. This figure reveals that the value of the threshold that maximizes the average MCC across folds is $\tau \approx 0573$. But more importantly, it seems that the "landscape" of the MCC around this value is relatively flat – in other words, as long as we do not pick a threshold that is too outlandishly low (or high!), the model would have a good performance. It is worth pausing for a minute and questioning *why* that is.

To do so, we can look at the distribution of probabilities returned by the NBC, which are presented in Figure 4.3. It appears that the NBC is often confident in its recommendations, with a bimodal distribution of probabilities. For this reason, small changes in the position of the threshold would only affect a very small number of instances, and consequently only have a small effect on the MCC and other statistics. If the distribution of probabilities returned by the NBC had been different, the shape of the learning curve may have been a lot more skewed.

Looking at Figure 4.3, it appears that changing the threshold is changing the proportion of false positives and negatives; it is worth investigating exactly how this happens. We can explore this behavior in Figure 4.4. The points where the PPV and NPC curves meet is a good first approximation of the threshold (the MCC is not picking this exact point, but this is an approximation nonetheless).

4.4.2. How good is the model?

After picking a threshold and seeing how it relates to the distribution of probabilities in the model output, we can have a look at the ROC and PR curves. They are presented in Figure 4.5. In both cases, we see that the model is behaving correctly (it is nearing the point in the graph corresponding to perfect classifications), and importantly, we can check that the variability between the different folds is low. The model also outperforms the no-skill classifier. Taken together, these results give us a strong confidence in the fact that our model with the threshold applied represents an improvement over the version without the threshold.

In a sense, the ROC and PR curves are another projection of the results from Figure 4.4: a good classifier makes credible recommendations for its positive class, while maintaining credible recommendations for the negative class. Looking at several ways to express the performance of the classifier is a good idea, as a good understanding of how reliable our predicitions are depends on our ability to appraise these different sources of error.

4.4.3. Optimizing the prior

update on NBC prior

how to update? also threshold

illustration of result

discuss imbalance again

4.4.4. Testing and visualizing the final model

As we are now considering that our model is adequately trained, we can apply it to the testing data we had set aside early in Chapter 2. Applying the trained model to this data provides a fair estimate of the expected model performance, and relaying this information to people who are going to use the model is important.

We are *not* applying the older versions of the model to the testing data, as we had decided against this. We had established the rule of "we pick the best model as the one with the highest validation MCC", and this is what we will stick to. To do otherwise would be the applied machine learning equivalent of p-hacking, as the question of "what to do in case a model with lower validation MCC had a better performance on the testing data?" would arise, and we do not want to start questioning our central decision this late in the process.

We can start by taking a look at the confusion matrix on the testing data:

$$\begin{pmatrix} 123 & 17 \\ 39 & 380 \end{pmatrix}$$

This is very promising! There are far more predictions on the diagonal (503) than outside of it (56), which suggests an accurate classifier. The MCC of this model is 0.75, its true-skill statistic is 0.716, and its positive and negative predictive values are respectively 0.879 and 0.907. In other words: this model is *extremely* good. The values of PPV and NPV in particular are important to report: they tell us that when the model predicts a positive or negative outcome, it is expected to be correct more than 9 out of 10 times.

The final predictions are shown in Figure 4.7. Although the range map is very similar to the one we produced by the end of Chapter 3, the small addition of an optimized threshold leads to a model that is overall a little more accurate. In **?@sec-bagging**, we will focus on the uncertainty associated to this prediction.

4.5. Conclusion

In this chapter, we have refined a model by adopting a principled approach to establishing hyper-parameters. This resulted in a final trained model, which we applied to produce the final prediction of the distribution of *Sitta whiteheadi*. In **?@sec-explanations**, we will start asking "why"? Specifically, we will see a series of tools to evaluate why the model was making a specific prediction at a specific place, and look at the relationship between the importance of variables for model performance and for actual predictions. But before we do this, we will spend time in **?@sec-bagging** to discuss the uncertainty that is part of this model, and how it can be communicated.

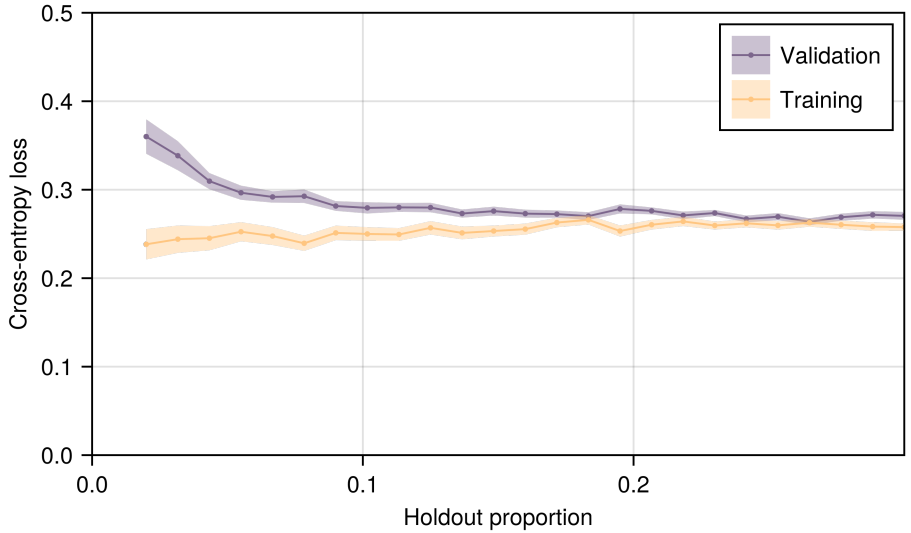


Figure 4.1.: Learning curve for the proportion of data used in hypothetical holdout cross-validation. The average MCC, as well as the 95% confidence interval around the MCC, are shown for 100 replicates. A higher holdout proportion indicates that fewer data are available for training. In practice, treating the cross-validation strategy as an hyper-parameter is an important step in obtaining a fair evaluation of the model performance.

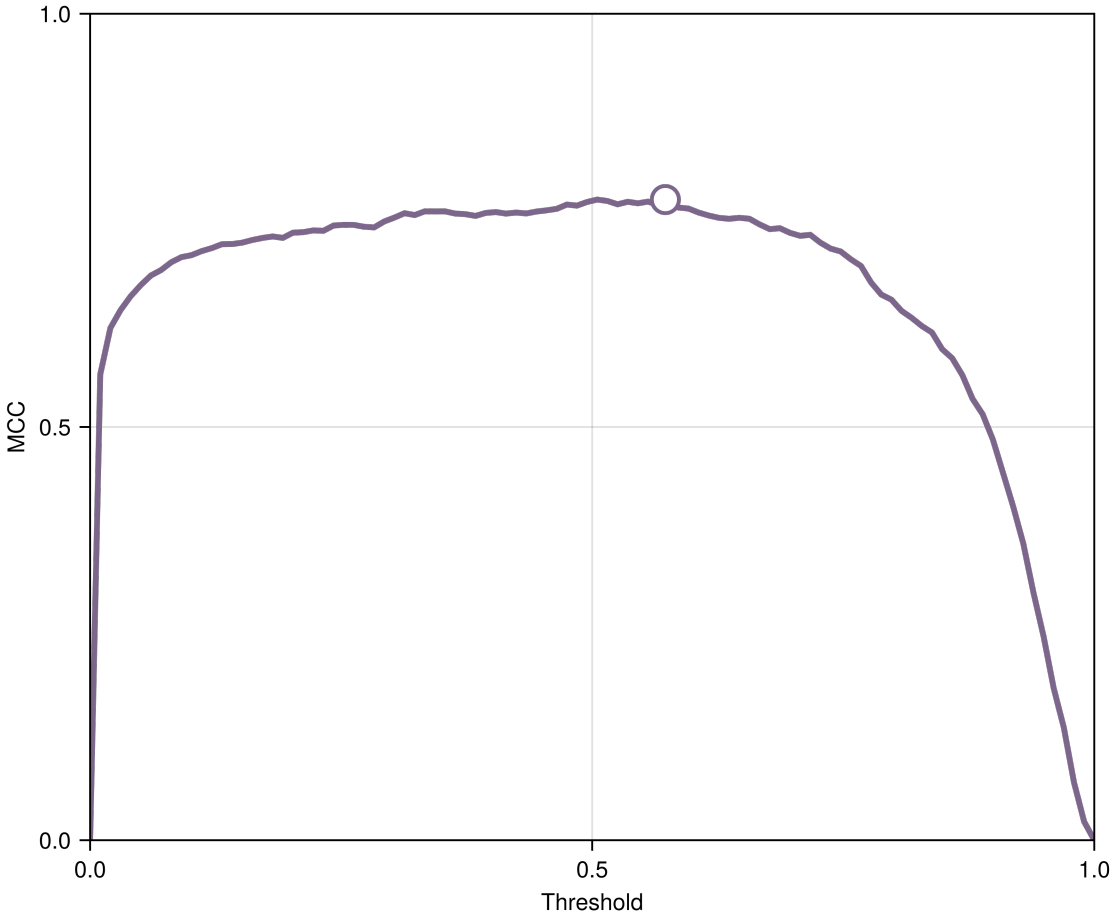


Figure 4.2.: Learning curve for the threshold of the NBC model. Note that the profile of the MCC with regards to the threshold is relatively flat. In other words, even picking a non-optimal value of the threshold would not necessarilly lead to a very bad model. Each grey line corresponds to a fold, and the blue line is the average.

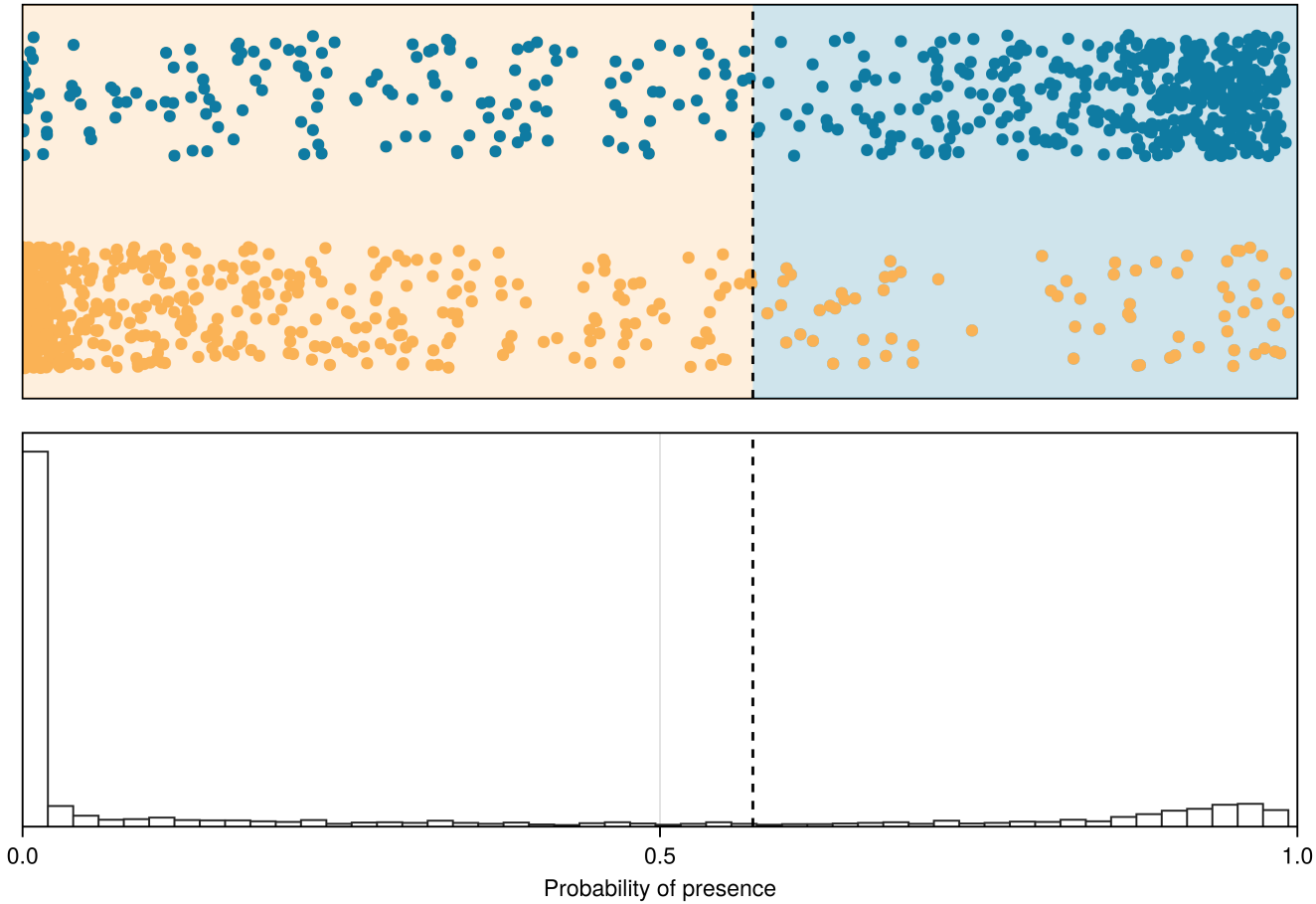


Figure 4.3.: Probabilities assigned to each pixel (bottom), color-coded by their value in the validation set (top scatterplots). The NBC is making a lot of recommendations very close to 0 or very close to 1, and for this reason, positioning the threshold anywhere in the middle of the range would give almost similar results in terms of the MCC.

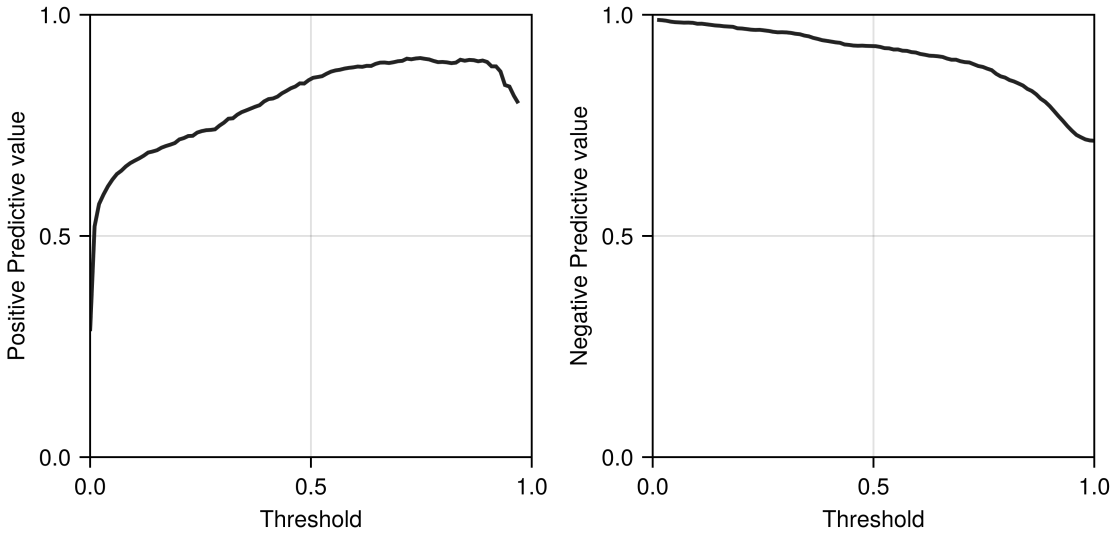


Figure 4.4.: Learning curve for the threshold of the NBC model, showing the PPV (solid line) and the NPV (dashed line). This figure shows how increasing the threshold leads to a better positive predictive value (we are more confident in the predicti class) at the cost of a loss in the negative predictive value (the pixels classified as negative are not meaningful). Essentially, moving the threshold (and indeed, tuning any other hyper-parameter) is a way to find a position in this space where the balance between errors maximizes the skill of our classifier.

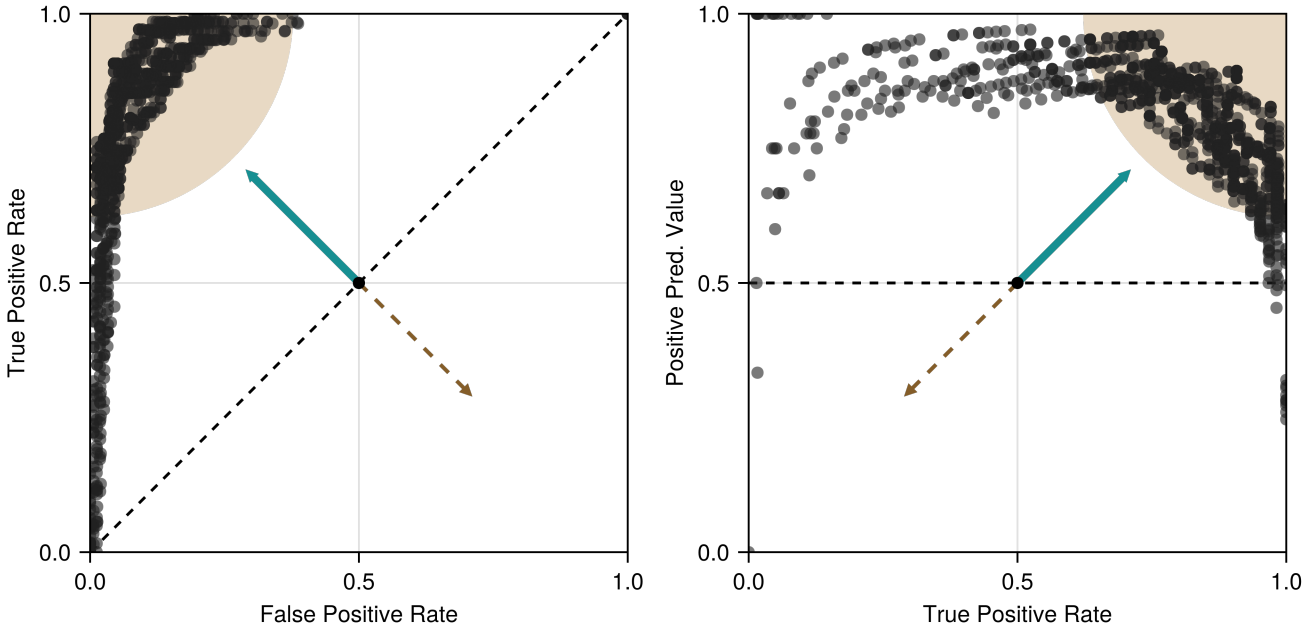


Figure 4.5.: ROC and PR curve for each fold, calculated on the validation datasets. The area highlighted in green corresponds to perfect classifiers, and the dashed line is the no-skill classifier. The solid arrow shows direction alongside which model performance increases in both cases.

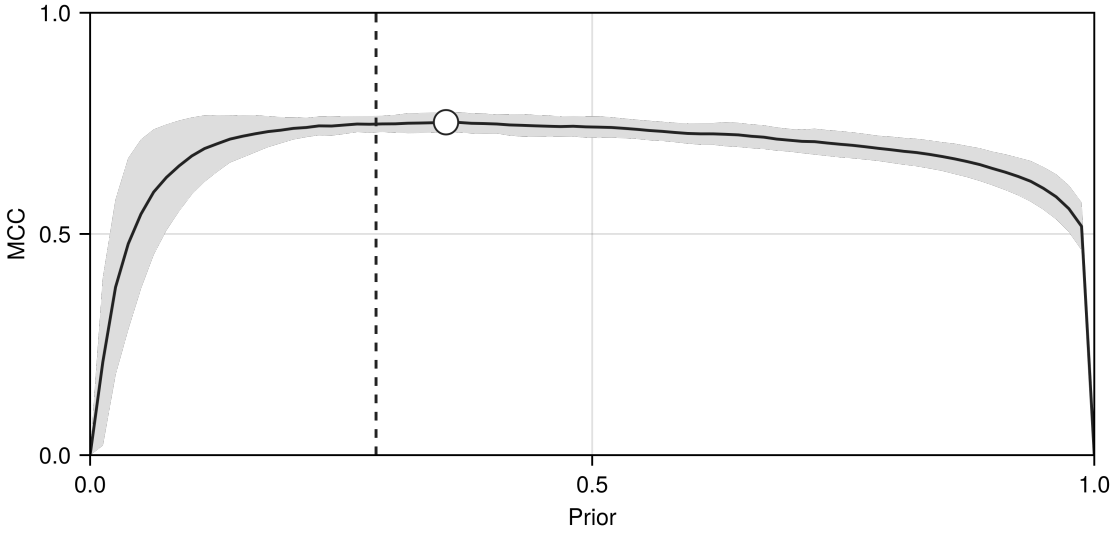


Figure 4.6.: Tuning of the NBC prior probability (see Equation 2.4) that a location is favorable to the presence of the species. As in Figure 4.2, the model was cross-validation on a number of priors increasing from 0 (presences will never be predicted) to 1 (presences will always be predicted). The vertical dashed line is the class balance in the dataset, showing that the optimal prior for this model is informed by the relative proportion of presences in the training data (this is the value of the prior we used by default in this entire chapter).

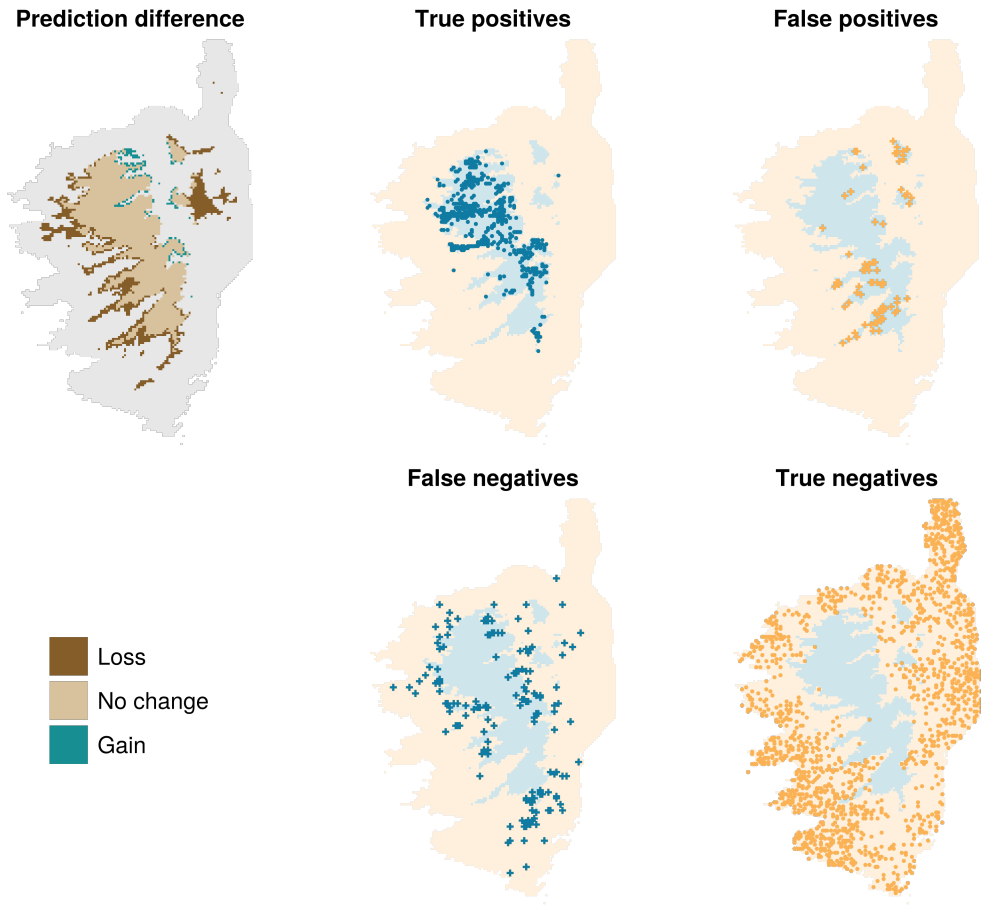


Figure 4.7.: Predicted range of *Sitta whiteheadi* (left) and associated bootstrap uncertainty (right; see Chapter 2). This prediction was made using the final trained model, including variable selection, transformations, and thresholding of the probability.

References

Becker, D.J., Albery, G.F., Sjodin, A.R., Poisot, T., Bergner, L.M., Chen, B., *et al.* (2022). Optimising predictive models to prioritise viral discovery in zoonotic reservoirs. *The Lancet Microbe*, 3, e625–e637.

Chicco, D. & Jurman, G. (2023). The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification. *BioData Mining*, 16.

Claesen, M. & De Moor, B. (2015). Hyperparameter search in machine learning.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 861–874.

Halligan, S., Altman, D.G. & Mallett, S. (2015). Disadvantages of using the area under the receiver operating characteristic curve to assess imaging tests: A discussion and proposal for an alternative approach. *European Radiology*, 25, 932–939.

Hanley, J.A. & McNeil, B.J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143, 29–36.

Huntington, D.E. & Lyrintzis, C.S. (1998). Improvements to and limitations of Latin hypercube sampling. *Probabilistic Engineering Mechanics*, 13, 245–253.

Jamieson, K. & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In: *Proceedings of the 19th international conference on artificial intelligence and statistics*, Proceedings of machine learning research (eds. Gretton, A. & Robert, C.C.). PMLR, Cadiz, Spain, pp. 240–248.

Jurman, G., Riccadonna, S. & Furlanello, C. (2012). A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. *PLoS ONE*, 7, e41882.

McKay, M.D., Beckman, R.J. & Conover, W.J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21, 239.

Pautasso, M. (2010). Worsening file-drawer problem in the abstracts of natural, medical and social science databases. *Scientometrics*, 85, 193–202.

Perkins, N.J. & Schisterman, E.F. (2005). The Youden Index and the Optimal Cut-Point Corrected for Measurement Error. *Biometrical Journal*, 47, 428–441.

Perkins, N.J. & Schisterman, E.F. (2006). The Inconsistency of "Optimal" Cutpoints Obtained using Two Criteria based on the Receiver Operating Characteristic Curve. *American Journal of Epidemiology*, 163, 670–675.

Poisot, T. (2023). Guidelines for the prediction of species interactions through binary classification. *Methods in Ecology and Evolution*, 14, 1333–1345.

Poisot, T., Ouellet, M.-A., Mollentze, N., Farrell, M.J., Becker, D.J., Brierley, L., *et al.* (2023). Network embedding

unveils the hidden interactions in the mammalian virome. *Patterns*, 4, 100738.

Saito, T. & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10, e0118432.

Unal, I. (2017). Defining an Optimal Cut-Point Value in ROC Analysis: An Alternative Approach. *Computational and Mathematical Methods in Medicine*, 2017, 1–14.

Yang, L. & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415, 295–316.

Youden, W.J. (1950). Index for rating diagnostic tests. *Cancer*, 3, 32–35.

Zhou, H. & Jakobsson, E. (2013). Predicting Protein-Protein Interaction by the Mirrortree Method: Possibilities and Limitations. *PLoS ONE*, 8, e81100.

A. Instructor notes

References

Abbood, A., Ullrich, A., Busche, R. & Ghozzi, S. (2020). EventEpi—A natural language processing framework for event-based surveillance. *PLOS Computational Biology*, 16, e1008277.

ALLOUCHE, O., TSOAR, A. & KADMON, R. (2006). Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). *Journal of Applied Ecology*, 43, 1223–1232.

Amarasinghe, K., Rodolfa, K.T., Lamba, H. & Ghani, R. (2023). Explainable machine learning for public policy: Use cases, gaps, and research directions. *Data & Policy*, 5.

Anderson, E. (1928). The problem of species in the northern blue flags, iris versicolor l. And iris virginica l. *Annals of the Missouri Botanical Garden*, 15, 241.

Barbet-Massin, M. & Jiguet, F. (2011). Back from a Predicted Climatic Extinction of an Island Endemic: A Future for the Corsican Nuthatch. *PLoS ONE*, 6, e18228.

Barbet-Massin, M., Jiguet, F., Albert, C.H. & Thuiller, W. (2012). Selecting pseudo-absences for species distribution models: how, where and how many? *Methods in Ecology and Evolution*, 3, 327–338.

Becker, D.J., Albery, G.F., Sjodin, A.R., Poisot, T., Bergner, L.M., Chen, B., *et al.* (2022). Optimising predictive models to prioritise viral discovery in zoonotic reservoirs. *The Lancet Microbe*, 3, e625–e637.

Beery, S., Cole, E., Parker, J., Perona, P. & Winner, K. (2021). Species distribution modeling for machine learning practitioners: A review. *ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS)*.

Berteaux, D. (2014). *Changements climatiques et biodiversité du québec*. Presses de l'Université du Québec.

Bezanson, J., Edelman, A., Karpinski, S. & Shah, V.B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59, 65–98.

Blaom, A., Kiraly, F., Lienart, T., Simillides, Y., Arenas, D. & Vollmer, S. (2020). MLJ: A julia package for composable machine learning. *Journal of Open Source Software*, 5, 2704.

Bodmer, W., Bailey, R.A., Charlesworth, B., Eyre-Walker, A., Farewell, V., Mead, A., *et al.* (2021). The outstanding scientist, R.A. Fisher: his views on eugenics and race. *Heredity*, 126, 565–576.

Booth, T.H. (2022). Checking bioclimatic variables that combine temperature and precipitation data before their use in species distribution models. *Austral Ecology*, 47, 1506–1514.
Chicco, D. & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21.
Chicco, D. & Jurman, G. (2023). The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification. *BioData Mining*, 16.
Chollet Ramampiandra, E., Scheidegger, A., Wydler, J. & Schuwirth, N. (2023). A comparison of machine learning and statistical species distribution models: Quantifying overfitting supports model interpretation. *Ecological Modelling*, 481, 110353.
Claesen, M. & De Moor, B. (2015). Hyperparameter search in machine learning.
Clapham, A.R., Raunkiaer, C., Gilbert-Carter, H., Tansley, A.G. & Fausboll. (1935). The life forms of plants and statistical plant geography. *The Journal of Ecology*, 23, 247.
De Marco, P. & Nóbrega, C.C. (2018). Evaluating collinearity effects on species distribution models: An approach based on virtual species simulation. *PLOS ONE*, 13, e0202403.
Deisenroth, M.P., Faisal, A.A. & Ong, C.S. (2020). Mathematics for machine learning.
Desai, J., Watson, D., Wang, V., Taddeo, M. & Floridi, L. (2022). The epistemological foundations of data science: a critical review. *Synthese*, 200.
Dietze, M. (2017). Ecological forecasting.
Dormann, C.F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., *et al.* (2012). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*, 36, 27–46.
Dornelas, M., Antão, L.H., Moyes, F., Bates, A.E., Magurran, A.E., Adam, D., *et al.* (2018). BioTIME: A database of biodiversity time series for the Anthropocene. *Global Ecology and Biogeography*, 27, 760–786.
Elith, J. & Leathwick, J.R. (2009). Species Distribution Models: Ecological Explanation and Prediction Across Space and Time. *Annual Review of Ecology, Evolution, and Systematics*, 40, 677–697.
Ellwood, E.R., Sessa, J.A., Abraham, J.K., Budden, A.E., Douglas, N., Guralnick, R., *et al.* (2019). Biodiversity Science and the Twenty-First Century Workforce. *BioScience*, 70, 119–121.
Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 861–874.
Fick, S.E. & Hijmans, R.J. (2017). WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, 37, 4302–4315.
Fisher, R.A. (1936). The Use Of Multiple Measurements In Taxonomic Problems. *Annals of Eugenics*, 7, 179–188.
Fox, E.W., Hill, R.A., Leibowitz, S.G., Olsen, A.R., Thornbrugh, D.J. & Weber, M.H. (2017). Assessing the

accuracy and stability of variable selection methods for random forest modeling in ecology. *Environmental Monitoring and Assessment*, 189.

Gorman, K.B., Williams, T.D. & Fraser, W.R. (2014). Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis). *PLoS ONE*, 9, e90081.

Graham, M.H. (2003). CONFRONTING MULTICOLLINEARITY IN ECOLOGICAL MULTIPLE REGRESSION. *Ecology*, 84, 2809–2815.

Guillera-Arroita, G., Lahoz-Monfort, J.J., Elith, J., Gordon, A., Kujala, H., Lentini, P.E., *et al.* (2015). Is my species distribution model fit for purpose? Matching data and models to applications. *Global Ecology and Biogeography*, 24, 276–292.

Halligan, S., Altman, D.G. & Mallett, S. (2015). Disadvantages of using the area under the receiver operating characteristic curve to assess imaging tests: A discussion and proposal for an alternative approach. *European Radiology*, 25, 932–939.

Hanberry, B.B., He, H.S. & Palik, B.J. (2012). Pseudoabsence Generation Strategies for Species Distribution Models. *PLoS ONE*, 7, e44486.

Hand, D.J. & Yu, K. (2001). Idiot's bayes: Not so stupid after all? *International Statistical Review / Revue Internationale de Statistique*, 69, 385.

Hanley, J.A. & McNeil, B.J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143, 29–36.

Horst, A.M., Hill, A.P. & Gorman, K.B. (2020). *Allisonhorst/palmerpenguins: v0.1.0*. Zenodo.

Howley, T., Madden, M.G., O'Connell, M.-L. & Ryder, A.G. (2005). The effect of principal component analysis on machine learning accuracy with high dimensional spectral data. Springer London, pp. 209–222.

Huntington, D.E. & Lyrintzis, C.S. (1998). Improvements to and limitations of Latin hypercube sampling. *Probabilistic Engineering Mechanics*, 13, 245–253.

Jamieson, K. & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In: *Proceedings of the 19th international conference on artificial intelligence and statistics*, Proceedings of machine learning research (eds. Gretton, A. & Robert, C.C.). PMLR, Cadiz, Spain, pp. 240–248.

Jurman, G., Riccadonna, S. & Furlanello, C. (2012). A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. *PLoS ONE*, 7, e41882.

Karger, D.N., Conrad, O., Böhner, J., Kawohl, T., Kreft, H., Soria-Auza, R.W., *et al.* (2017). Climatologies at high resolution for the earth's land surface areas. *Scientific Data*, 4.

Kaufman, S., Rosset, S. & Perlich, C. (2011). Leakage in data mining. *Proceedings of the 17th ACM SIGKDD*

international conference on Knowledge discovery and data mining.

Kessy, A., Lewin, A. & Strimmer, K. (2018). Optimal Whitening and Decorrelation. *The American Statistician*, 72, 309–314.

Koivunen, A.C. & Kostinski, A.B. (1999). The Feasibility of Data Whitening to Improve Performance of Weather Radar. *Journal of Applied Meteorology*, 38, 741–749.

Kupervasser, O. (2014). The mysterious optimality of Naive Bayes: Estimation of the probability in the system of "classifiers". *Pattern Recognition and Image Analysis*, 24, 1–10.

Legendre, P. & Legendre, L. (2012). *Numerical ecology*. Developments in environmental modelling. Third English edition. Elsevier, Oxford, UK.

Leroy, B., Delsol, R., Hugueny, B., Meynard, C.N., Barhoumi, C., Barbet-Massin, M., *et al.* (2018). Without quality presence–absence data, discrimination metrics such as TSS can be misleading measures of model performance. *Journal of Biogeography*, 45, 1994–2002.

McKay, M.D., Beckman, R.J. & Conover, W.J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21, 239.

Murtaugh, P.A. (2009). Performance of several variable-selection methods applied to real ecological data. *Ecology Letters*, 12, 1061–1068.

Nisbet, R., Miner, G., Yale, K., Elder, J.F. & Peterson, A.F. (2018). *Handbook of statistical analysis and data mining applications*. Second edition. Academic Press, London.

Pautasso, M. (2010). Worsening file-drawer problem in the abstracts of natural, medical and social science databases. *Scientometrics*, 85, 193–202.

Pearson, K. (1901). LIII. *On lines and planes of closest fit to systems of points in space*. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2, 559–572.

Perkins, N.J. & Schisterman, E.F. (2005). The Youden Index and the Optimal Cut-Point Corrected for Measurement Error. *Biometrical Journal*, 47, 428–441.

Perkins, N.J. & Schisterman, E.F. (2006). The Inconsistency of "Optimal" Cutpoints Obtained using Two Criteria based on the Receiver Operating Characteristic Curve. *American Journal of Epidemiology*, 163, 670–675.

Perl, R.G.B., Avidor, E., Roll, U., Malka, Y., Geffen, E. & Gafny, S. (2022). Using eDNA presence/non-detection data to characterize the abiotic and biotic habitat requirements of a rare, elusive amphibian. *Environmental DNA*, 4, 642–653.

Peterson, A.T., Asase, A., Canhos, D., Souza, S. de & Wieczorek, J. (2018). Data leakage and loss in biodiversity informatics. *Biodiversity Data Journal*, 6.

Petitpierre, B., Broennimann, O., Kueffer, C., Daehler, C. & Guisan, A. (2016). Selecting predictors to maximize the transferability of species distribution models: lessons from cross-continental plant invasions. *Global Ecology and Biogeography*, 26, 275–287.

Poisot, T. (2023). Guidelines for the prediction of species interactions through binary classification. *Methods in Ecology and Evolution*, 14, 1333–1345.

Poisot, T., Ouellet, M.-A., Mollentze, N., Farrell, M.J., Becker, D.J., Brierley, L., *et al.* (2023). Network embedding unveils the hidden interactions in the mammalian virome. *Patterns*, 4, 100738.

Powers, D.M.W. (2020). Evaluation: From precision, recall and f-measure to ROC, informedness, markedness and correlation. *arXiv*.

Runghen, R., Stouffer, D.B. & Dalla Riva, G.V. (2022). Exploiting node metadata to predict interactions in bipartite networks using graph embedding and neural networks. *Royal Society Open Science*, 9.

Saito, T. & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10, e0118432.

Schölkopf, B., Smola, A. & Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10, 1299–1319.

Smith, M.L., Ruffley, M., Espíndola, A., Tank, D.C., Sullivan, J. & Carstens, B.C. (2017). Demographic model selection using random forests and the site frequency spectrum. *Molecular Ecology*, 26, 4562–4573.

Stock, A., Gregr, E.J. & Chan, K.M.A. (2023). Data leakage jeopardizes ecological applications of machine learning. *Nature Ecology & Evolution*.

Sulmont, E., Patitsas, E. & Cooperstock, J.R. (2019). Can you teach me to machine learn? *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*.

Thuiller, W., Araújo, M.B. & Lavorel, S. (2004). Do we need land-cover data to model species distributions in Europe? *Journal of Biogeography*, 31, 353–361.

Tipping, M.E. & Bishop, C.M. (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61, 611–622.

Tredennick, A.T., Hooker, G., Ellner, S.P. & Adler, P.B. (2021). A practical guide to selecting models for exploration, inference, and prediction in ecology. *Ecology*, 102.

Tuia, D., Kellenberger, B., Beery, S., Costelloe, B.R., Zuffi, S., Risse, B., *et al.* (2022). Perspectives in machine learning for wildlife conservation. *Nature Communications*, 13, 792.

Unal, I. (2017). Defining an Optimal Cut-Point Value in ROC Analysis: An Alternative Approach. *Computational and Mathematical Methods in Medicine*, 2017, 1–14.

Unwin, A. & Kleinman, K. (2021). The Iris Data Set: In Search of the Source of *Virginica*. *Significance*, 18, 26–29.
Vasseur, D.A. & Yodzis, P. (2004). THE COLOR OF ENVIRONMENTAL NOISE. *Ecology*, 85, 1146–1152.
Watt, J., Borhani, R. & Katsaggelos, A. (2020). Machine learning refined.
Whittaker, R.H. (1962). Classification of natural communities. *The Botanical Review*, 28, 1–239.
WHITTINGHAM, M.J., STEPHENS, P.A., BRADBURY, R.B. & FRECKLETON, R.P. (2006). Why do we still use stepwise modelling in ecology and behaviour? *Journal of Animal Ecology*, 75, 1182–1189.
Yang, L. & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415, 295–316.
Yau, N. (2015). Visualize this.
Youden, W.J. (1950). Index for rating diagnostic tests. *Cancer*, 3, 32–35.
Zhou, H. & Jakobsson, E. (2013). Predicting Protein-Protein Interaction by the Mirrortree Method: Possibilities and Limitations. *PLoS ONE*, 8, e81100.

Index