

Machine Learning for Biodiversity Scientists

An opinionated primer

Timothée Poisot

2024-10-19

Table of contents

Preface	1
1. Introduction	3
1.1. Core concepts in data science	5
1.2. An overview of the content	5
1.3. A note on colors	6
1.4. Some rules about this book	8
References	11
2. Clustering	15
2.1. A digression: which birds are red?	15
2.2. The problem: classifying pixels from an image	17
2.3. The theory behind k -means clustering	21
2.4. Application: optimal clustering of the satellite image data	26
2.5. Is k -means a model?	32
2.6. Conclusion	34
References	34
3. Gradient descent	37
3.1. A digression: what is a trained model?	37
3.2. The problem: how many interactions in a food web?	38
3.3. Gradient descent	39
3.4. Application: how many links are in a food web?	45
3.5. A note on regularization	52
3.6. Conclusion	52
References	53

Table of contents

4. Cross-validation	55
4.1. How can we split a dataset?	55
4.2. The problem: cherry blossom phenology	59
4.3. Strategies to split data	60
4.4. Application: when do cherry blossom bloom?	65
4.5. Conclusion	70
References	72
5. Supervised classification	75
5.1. The problem: distribution of an endemic species	75
5.2. What is classification?	77
5.3. The Naive Bayes Classifier	82
5.4. Application: a baseline model of the Corsican nuthatch	86
5.5. Conclusion	93
Appendices	95
References	95
A. Instructor notes	97
References	99
Index	105

List of Figures

- | | |
|---|----|
| 2.1. The Landsat 8 data are combined into the “Natural Color” image, in which the red, green, and blue bands are mapped to their respective channels (left). The other composites are different composites (SWIR1, NIR, red; NIR, red, green; SWIR2, SWIR1, red), highlighting how choices of data representation are likely to affect what areas of the landscape will pop up. Note that the true-color composite is slightly distorted compared to the colors of the landscape we expect; this is because natural colors are difficult to reproduce accurately. | 19 |
| 2.2. The pixels acquired from Landsat 8 exist in a space with many different dimensions (one for each band). Because we are interested in a landscape classification based on water and vegetation data, we use the NDVI, NDMI, NBR, and NDWI combinations of bands. These are <i>derived</i> data, and represent the creation of new features from the raw data. Darker colors indicate more pixels in this bin. | 22 |
| 2.3. After iterating the k-means algorithm, we obtain a classification for every pixel in the landscape. This classification is based on the values of NDVI, NDMI, and NDWI indices, and therefore groups pixels based on specific assumptions about vegetation and stress. This clustering was produced using $k = 3$, i.e. we want to see what the landscape would look like when divided into three categories. | 27 |
| 2.4. Results of running the k -means algorithm ten times for each number of clusters between 3 and 10. The average Davies-Bouldin and cost are reported, as well as the standard deviation. As expected, the total cost decreases with more clusters, but this is not necessarily the sign of a better clustering. | 30 |
| 2.5. Results of the landscape clustering with $k=4$ clusters. This number of clusters gives us a good separation between different groups of pixels, and seems to capture features of the landscape as revealed with the false-color composites. | 31 |
| 2.6. Visualisation of the clustering output as a function of the NDVI and NBR values. Note that the limits between the clusters are lines (planes), and that each cluster covers about the same volume in the space of parameters. | 33 |

List of Figures

3.1.	We have assumed that the relationship between L and S could be represented by $L \approx b \times S^a$, which gave us a reason to take the natural log of both variables. On this figure, we see that the relationship between the logs look linear, which means that linear regression has a good chance of estimating the values of the parameters.	40
3.2.	This figures shows the change in the loss for the training and testing dataset. As the two curves converge on low values at the same rate, this suggests that the model is not over-fitting, and is therefore suitable for use.	48
3.3.	This figure shows the change in the parameters values over time. Note that the change is very large initially, because we make large steps when the gradient is strong. The rate of change gets much lower as we get nearer to the “correct” value.	50
3.4.	Overview of the fitted model. The residuals (top panel) are mostly centered around 0, which suggests little bias towards over/under predicting interactions. The red line (based on the optimal coefficients) goes through the points, and indicates a rather good fit of the model.	51
4.1.	Relationship between temperature in March and day of blooming.	61
4.2.	Illustration of different folds on a timeseries.	64
4.3.	Model performance according to three loss functions.	67
4.4.	Fit of the final model, after training on all instances.	69
4.5.	Model predictions on the testing data.	71
5.1.	This figures show the separability of the presences (orange) and pseudo-absences (grey) on the temperature and precipitation dimensions.	79
5.2.	Overview of the scores for the Matthew’s correlation coefficient, as well as the positive and negative predictive values.	88
5.3.	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.	90
5.4.	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 <i>Sitta whiteheadi</i> , but would definitely stand to be improved, possibly by accounting for more features.	92

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.

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.

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.

Preface

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).

1. Introduction

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 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 Chapter 2, 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 Chapter 3, we revisit well-trodden statistical ground, by fitting a linear model to linear data, but using 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 Chapter 4, 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 5, 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.

1. Introduction

In **?@sec-predictors**, 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 **?@sec-leakage**, 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 **?@sec-tuning**, 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 our 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	

1.3. A note on colors

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

In addition, there are three important color *palettes*. Information that is *sequential* in 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 takes 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 corresponds 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.

1. Introduction

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 spend 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!

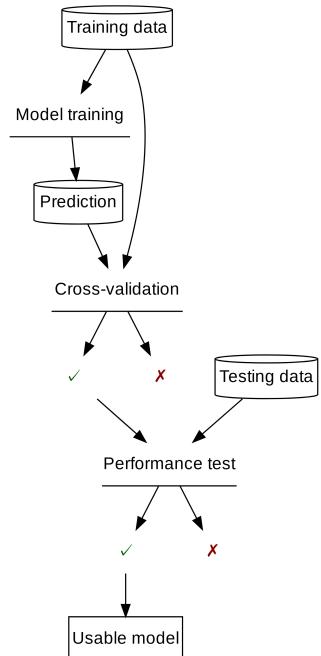
The most broadly known alternative to the *iris* dataset is *penguins*, which was collected by ecologists (Gorman *et al.* 2014), and published as a standard dataset (Horst *et al.* 2020) so that we can train students without engaging with the “legacy” of eugenicists. The *penguins* dataset is also genuinely 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*.

1. Introduction

- 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 training 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 then applied to an independent testing dataset, and the evaluation of the performance determines whether it will be used.

2. Clustering

As we mentioned in the introduction, a core idea of data science is that things that look the same (in that, when described with data, they resemble one another) are likely to be the same. Although this sounds like a simplifying assumption, this can provide the basis for approaches in which we *create* groups in data that have no labels. This task is called clustering: we seek to add a *label* to each observation, in order to form groups, and the data we work from do *not* have a label that we can use to train a model. In this chapter, we will explore the *k*-means algorithm for clustering, and illustrate how it can be used in practice.

2.1. A digression: which birds are red?

Before diving in, it is a good idea to ponder a simple case. We can divide everything in just two categories: things with red feathers, and things without red feathers. An example of a thing with red feathers is the Northern Cardinal (*Cardinalis cardinalis*), and things without red feathers are the iMac G3, Haydn's string quartets, and of course the Northern Cardinal (*Cardinalis cardinalis*).

See, biodiversity data science is complicated, because it tends to rely on the assumption that we can categorize the natural world, and the natural world (mostly in response to natural selection) comes up with ways to be, well, diverse and hard to categorize. In the Northern Cardinal, this is shown in males having red feathers, and females having mostly brown feathers. Before moving forward, we need to consider ways to solve this issue, as this issue will come up *all the time*.

The first mistake we have made is that the scope of objects we want to classify, which we will describe as the "domain" of our classification, is much too broad: there are few legitimate applications where we will have a dataset with Northern Cardinals, iMac G3s, and Haydn's string quartets. Picking a reasonable universe of

2. Clustering

classes would have solved our problem a little. For example, among the things that do not have red feathers are the Mourning Dove, the Kentucky Warbler, and the House Sparrow.

The second mistake that we have made is improperly defining our classes; bird species exhibit sexual dimorphism (not in an interesting way, like wrasses, but let's give them some credit for trying). Assuming that there is such a thing as a Northern Cardinal is not necessarily a reasonable assumption! And yet, the assumption that a single label is a valid representation of non-monomorphic populations is a surprisingly common one, with actual consequences for the performance of image classification algorithms (Luccioni & Rolnick 2023). This assumption reveals a lot about our biases: male specimens are over-represented in museum collections, for example (Cooper *et al.* 2019). In a lot of species, we would need to split the taxonomic unit into multiple groups in order to adequately describe them.

The third mistake we have made is using predictors that are too vague. The “presence of red feathers” is not a predictor that can easily discriminate between the Northern Cardinal (yes for males, sometimes for females), the House Finch (a little for males, no for females), and the Red-Winged Black Bird (a little for males, no for females). In fact, it cannot really capture the difference between red feathers for the male House Finch (head and breast) and the male Red Winged Black Bird (wings, as the name suggests).

The final mistake we have made is in assuming that “red” is relevant as a predictor. In a wonderful paper, Cooney *et al.* (2022) have converted the color of birds into a bird-relevant colorimetric space, revealing a clear latitudinal trend in the ways bird colors, as perceived by other birds, are distributed. This analysis, incidentally, splits all species into males and females. The use of a color space that accounts for the way colors are perceived is a fantastic example of why data science puts domain knowledge front and center.

Deciding which variables are going to be accounted for, how the labels will be defined, and what is considered to be within or outside the scope of the classification problem is *difficult*. It requires domain knowledge (you must know a few things about birds in order to establish criteria to classify birds), and knowledge of how the classification methods operate (in order to have just the right amount of overlap between features in order to provide meaningful estimates of distance).

2.2. The problem: classifying pixels from an image

Throughout this chapter, we will work on a single image – we may initially balk at the idea that an image is data, but it is! Specifically, an image is a series of instances (the pixels), each described by their position in a multidimensional colorimetric space. Greyscale images have one dimension, and images in color will have three: their red, green, and blue channels. Not only are images data, this specific dataset is going to be far larger than many of the datasets we will work on in practice: the number of pixels we work with is given by the product of the width, height, and depth of the image!

In fact, we are going to use an image with many dimensions: the data in this chapter are coming from a Landsat 8 scene (Vermote *et al.* 2016), for which we have access to many different bands.

Table 2.1.: Overview of the bands in a Landsat 8 scene. The data from this chapter were downloaded using the Google Earth Engine API, and are a composite of cloud-free and shadow-free pixels between the months of April and October of 2017. Note that we are spanning a wide range of seasonal conditions!

Band	Measure	Notes
1	Aerosol	Good proxy for Chl. in oceans
2	Visible blue	
3	Visible green	
4	Visible red	
5	Near-infrared (NIR)	Reflected by healthy plants
6, 7	Short wavelength IR (SWIR 1 and 2)	Good at differentiating wet earth and dry earth
8	Panchromatic	High-resolution monochrome
9	Cirrus band	Can pick up high and thin clouds
10, 11	Thermal infrared	

By using the data present in the channels, we can reconstruct an approximation of what the landscape looked like (by using the red, green, and blue channels).

2. Clustering

Or can we?

If we were to invent a time machine, and go stand directly under Landsat 8 at the exact center of this scene, and look around, what would we see? We would see colors, and they would admit a representation as a three-dimensional vector of red, green, and blue. But we would see so much more than that! And even if we were to stand within a pixel, we would see a *lot* of colors. And texture. And depth. We would see something entirely different from this map; and we would be able to draw a lot more inferences about our surroundings than what is possible by knowing the average color of a 25x25 meters pixel. But just like we can get more information that Landsat 8, so too can Landsat 8 out-sense us when it comes to getting information. In the same way that we can extract a natural color composite out of the different channels, we can extract a fake color one to highlight differences in the landscape.

In Figure 2.1, we compare the natural color reconstruction (top) to a false color composite. All of the panels in Figure 2.1 represent the same physical place at the same moment in time; but through them, we are looking at this place with very different purposes. This is not an idle observation, but a core notion in data science: *what we measure defines what we can see*. In order to tell something ecologically meaningful about this place, we need to look at it in the “right” way. Of course, although remote sensing offers a promising way to collect data for biodiversity monitoring at scale (Gonzalez *et al.* 2023), there is no guarantee that it will be the right approach for all problems. More (fancier) data is not necessarily awlays adequate, and just because an equipment is very expensive does not mean it will provide the information we *need*.

We will revisit the issue of variable selection and feature engineering in
?`@sec-predictors`.

So far, we have looked at this area by combining the raw data. Depending on the question we have in mind, they may not be the *right* data. In fact, they may not hold information that is relevant to our question *at all*; or worse, they can hold more noise than signal. The area we will work on in this chapter is a very small crop of a Landsat 8 scene, showing a median of the images with no cloud, for the year 2017, at a 25m resolution, for the southernmost part of Corsica. This is an interesting area because it has a high variety of environments: large bodies of water, forested areas, urban areas, and at least one airport.

But can we classify these different environments starting in an ecologically relevant way? Based on our knowledge of plants, we can start thinking about the question of splitting the landscape into classes in a different way. Specifically, “can we guess that a pixel contains plants?”, and “can we guess at how much water there is in a pixel?”. During the summer of 2017, a large forest fire took place, about 10 km north of Bonifacio; this

2.2. The problem: classifying pixels from an image

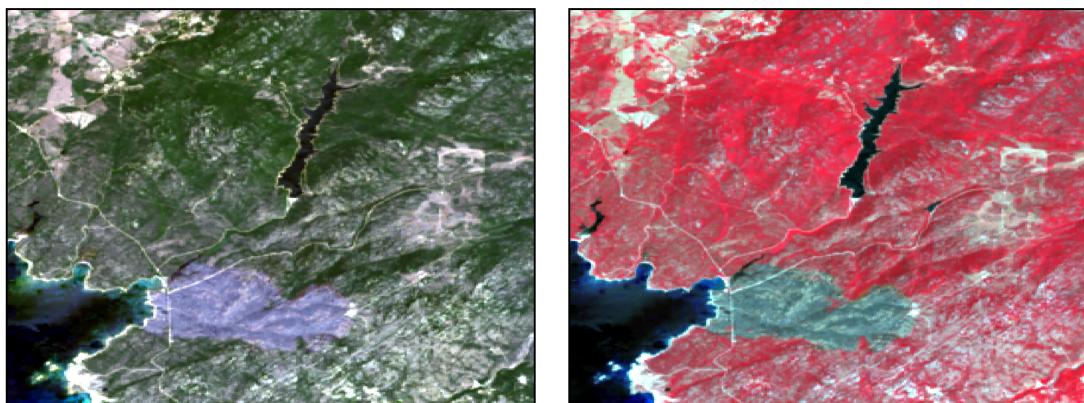
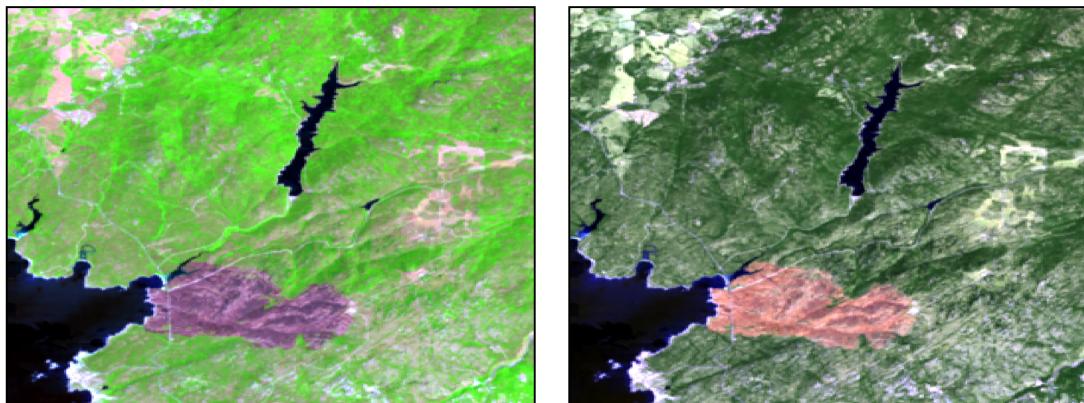


Figure 2.1.: The Landsat 8 data are combined into the “Natural Color” image, in which the red, green, and blue bands are mapped to their respective channels (left). The other composites are different composites (SWIR1, NIR, red; NIR, red, green; SWIR2, SWIR1, red), highlighting how choices of data representation are likely to affect what areas of the landscape will pop up. Note that the true-color composite is slightly distorted compared to the colors of the landscape we expect; this is because natural colors are difficult to reproduce accurately.



2. Clustering

is a strong incentive to also ask the question, “can we figure out which area burned?”. Thankfully, ecologists, whose hobbies include (i) guesswork and (ii) plants, have ways to answer these questions rather accurately.

One way to do this is to calculate the normalized difference vegetation index, or NDVI (Kennedy & Burbach 2020). NDVI is derived from the band data (NIR and Red), and there is an adequate heuristic using it to make a difference between vegetation, barren soil, and water. NDVI is a normalized difference index, which is a function of the form $f(x, y) = (x - y)/(x + y)$. Because plants are immediately tied to water, we can also consider the NDWI (water; Green and NIR), NDMI (moisture; NIR and SWIR1), and the NBR (normalized burn ratio; NIR and SWIR2) dimensions: taken together, these information will represent every pixel in a three-dimensional space, telling us whether there are plants (NDVI), whether they are stressed (NDMI), whether this pixel is a water body (NDWI), and whether this pixel is recovering following a fire (NBR). Roy *et al.* (2006) have challenged the idea that the NBR is relevant immediately post-fire, but negative NBR values can suggest re-growth after a fire.

Normalized differences indices are really fascinating measures to apply when you have two different quantities u and w . The formula for the normalized difference index is

$$\frac{u - w}{u + w}$$

or in other words, is the difference between these two quantities *relatively* larger than their sum. If $u = w$, their normalized difference is 0. If $w = 0$, or $u = 0$, their normalized difference is 1. Normalized differences are good at measuring the *over* or *under* representation of quantities you can reasonably add (“reasonably” meaning that you do not have strong statistical or biological arguments against). For example, Dansereau *et al.* (2024) used the normalized difference of different measures of food web structure to identify areas in space where different types of competition for food dominated ecological communities.

We can look at the pairwise relationships between these derived measures Figure 2.2. For example, NDMI values around -0.1 are [low-canopy cover with low water stress](#); NDVI values from 0.2 to 0.5 are good candidates for moderately dense crops. Notice that there is a strong (linear) relationship between NDVI and NDMI. Indeed, none of these indices are really independent; this implies that they are likely to be more informative taken together than when looking at them one at a time (Zheng *et al.* 2021). Indeed, urban area tend to have

2.3. The theory behind k-means clustering

high values of the NDWI, which makes the specific task of looking for swimming pools (for mosquito control) more challenging than it sounds (McFeeters 2013).

By picking these four transformed values, instead of simply looking at the clustering of all the bands in the raw data, we are starting to refine what the algorithm sees, through the lens of what we know (or guess!) to be important about the system. With these data in hand, we can start building a classification algorithm.

2.3. The theory behind k-means clustering

In order to understand the theory underlying k -means, we will work backwards from its output. As a method for clustering, k -means will return a vector of *class memberships*, which is to say, a list that maps each observation (pixel, in our case) to a class (tentatively, a cohesive landscape unit). What this means is that k -means is a transformation, taking as its input a vector with (in our case) four dimensions (NDVI, NDMI, NDWI, NBR), and returning a scalar (an integer, even!), giving the class to which this pixel belongs. Pixels only belongs to one class. These are the input and output of our blackbox, and now we can start figuring out its internals.

2.3.1. Inputs and parameters

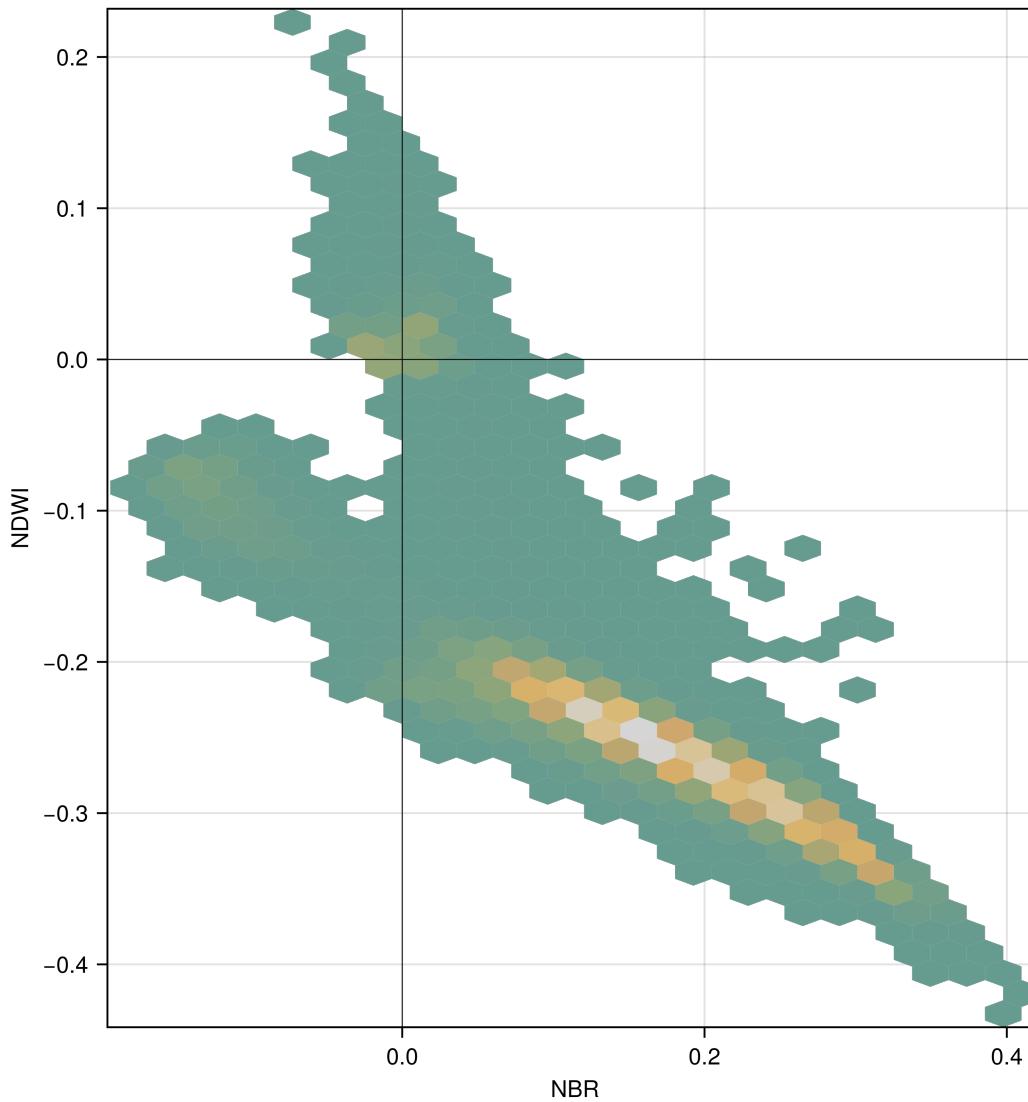
In k -means, a set of observations \mathbf{x}_i are assigned to a set of classes \mathbf{C} , also called the clusters. All \mathbf{x}_i are vectors with the same dimension (we will call it f , for *features*), and we can think of our observations as a matrix of features \mathbf{X} of size (f, n) , with f features and n observations (the columns of this matrix).

The number of classes of \mathbf{C} is $|\mathbf{C}| = k$, and k is an hyper-parameter of the model, as it needs to be fixed before we start running the algorithm. Each class is defined by its centroid, a vector \mathbf{c} with f dimensions (*i.e.* the centroid corresponds to a potential “idealized” observation of this class in the space of the features), which k -means progressively refines.

Throughout this book, we will use \mathbf{X} to note the matrix of features, and \mathbf{y} to note the vector of labels. Instances are columns of the matrix of features, and noted \mathbf{x}_i . When we need to represent a vector \mathbf{v} as a row vector, it will be noted \mathbf{v}^\top .

2. Clustering

Figure 2.2.: The pixels acquired from Landsat 8 exist in a space with many different dimensions (one for each band). Because we are interested in a landscape classification based on water and vegetation data, we use the NDVI, NDMI, NBR, and NDWI combinations of bands. These are *derived* data, and represent the creation of new features from the raw data. Darker colors indicate more pixels in this bin.



2.3.2. Assigning instances to classes

Instances are assigned to the class for which the distance between themselves and the centroid of this class is lower than the distance between themselves and the centroid of any other class. To phrase it differently, the class membership of an instance \mathbf{x}_i is given by

$$\operatorname{argmin}_j \|\mathbf{x}_i - \mathbf{c}_j\|_2 , \quad (2.1)$$

The correct distance measure to use depends on what is appropriate for the data! See for example Dove *et al.* (2023) for time-series, or Legendre & Gauthier (2014) for community composition data.

which is the value of j that minimizes the L^2 norm ($\|\cdot\|_2$, the Euclidean distance) between the instance and the centroid; argmin_j is the function returning the value of j that minimizes its argument. For example, $\operatorname{argmin}(0.2, 0.8, 0.0)$ is 3, as the third argument is the smallest. There exists an argmax function, which works in the same way.

2.3.3. Optimizing the centroids

Of course, what we really care about is the assignment of *all* instances to the classes. For this reason, the configuration (the disposition of the centroids) that solves our specific problem is the one that leads to the lowest possible variance within the clusters. As it turns out, it is not that difficult to go from Equation 2.1 to a solution for the entire problem: we simply have to sum over all points!

This leads to a measure of the variance, which we want to minimize, expressed as

$$\sum_{i=1}^k \sum_{\mathbf{x} \in \mathcal{C}_i} \|\mathbf{x} - \mathbf{c}_i\|_2 . \quad (2.2)$$

The part that is non-trivial is now to decide on the value of \mathbf{c} for each class. This is the heart of the k -means algorithm. From Equation 2.1, we have a criteria to decide to which class each instance belongs. Of course, there is nothing that prevents us from using this in the opposite direction, to define the instance by the points that form it! In this approach, the membership of class \mathcal{C}_j is the list of points that satisfy the condition in

2. Clustering

Equation 2.1. But there is no guarantee that the *current* position of \mathbf{c}_j in the middle of all of these points is optimal, *i.e.* that it minimizes the within-class variance.

This is easily achieved, however. To ensure that this is the case, we can re-define the value of \mathbf{c}_j as

$$\mathbf{c}_j = \frac{1}{|\mathbf{C}_j|} \sum \mathbf{C}_j, \quad (2.3)$$

where $|\cdot|$ is the cardinality of (number of instances in) \mathbf{C}_j , and $\sum \mathbf{C}_j$ is the sum of each feature in \mathbf{C}_j . To put it plainly: we update the centroid of \mathbf{C}_j so that it takes, for each feature, the average value of all the instances that form \mathbf{C}_j .

2.3.4. Updating the classes

Repeating a step multiple times in a row is called an iterative process, and we will see a *lot* of them. Chapter 3 relies almost entirely on iteration, in fact.

Once we have applied Equation 2.3 to all classes, there is a good chance that we have moved the centroids in a way that moved them away from some of the points, and closer to others: the membership of the instances has likely changed. Therefore, we need to re-start the process again, in an iterative way.

But until when?

Finding the optimal solution for a set of points is an NP-hard problem (Aloise *et al.* 2009), which means that we will need to rely on a little bit of luck, or a whole lot of time. The simplest way to deal with iterative processes is to let them run for a long time, as after a little while they should converge onto an optimum (here, a set of centroids for which the variance is as good as it gets), and hope that this optimum is *global* and not *local*.

A global optimum is easy to define: it is the state of the solution that gives the best possible result. For this specific problem, a global optimum means that there are no other combinations of centroids that give a lower variance. A local optimum is a little bit more subtle: it means that we have found a combination of centroids that we cannot improve without first making the variance worse. Because the algorithm as we have introduced it in the previous sections is *greedy*, in that it makes the moves that give the best short-term improvement, it will not provide a solution that temporarily makes the variance higher, and therefore is susceptible to being trapped in a local optimum.

2.3. The theory behind k-means clustering

In order to get the best possible solution, it is therefore common to run k -means multiple times for a given k , and to pick the positions of the centroids that give the best overall fit.

2.3.5. Identification of the optimal number of clusters

One question that is left un-answered is the value of k . How do we decide on the number of clusters?

There are two solutions here. One is to have an *a priori* knowledge of the number of classes. For example, if the purpose of clustering is to create groups for some specific task, there might be an upper/lower bound to the number of tasks you are willing to consider. The other solution is to run the algorithm in a way that optimizes the number of clusters for us.

This second solution turns out to be rather simple with k -means. We need to change the value of k , run it on the same dataset several times, and then pick the solution that was *optimal*. But this is not trivial. Simply using Equation 2.2 would lead to always preferring many clusters. After all, each point in its own cluster would get a pretty low variance!

For this reason, we use measures of optimality that are a little more refined. One of them is the Davies & Bouldin (1979) method, which is built around a simple idea: an assignment of instances to clusters is good if the instances within a cluster are not too far away from the centroids, and the centroids are as far away from one another as possible.

The Davies-Bouldin measure is striking in its simplicity. From a series of points and their assigned clusters, we only need to compute two things. The first is a vector \mathbf{s} , which holds the average distance between the points and their centroids (this is the $\|\mathbf{x}_i - \mathbf{c}_j\|_2$ term in Equation 2.1, so this measure still relates directly to the variance); the second is a matrix \mathbf{M} , which measures the distances *between* the centroids.

These two information are combined in a matrix \mathbf{R} , wherein $\mathbf{R}_{ij} = (s_i + s_j)/\mathbf{M}_{ij}$. The interpretation of this term is quite simply: is the average distance *within* clusters i and j much larger compared to the distance *between* these clusters. This is, in a sense, a measure of the stress that these two clusters impose on the entire system. In order to turn this matrix into a single value, we calculate the maximum value (ignoring the diagonal!) for each row: this is a measure of the *maximal* amount of stress in which a cluster is involved. By averaging these

We will spend a lot more time on the proper technique to optimize (“tune”) the hyperparameters of a model in [?@sec-tuning](#).

2. Clustering

values across all clusters, we have a measure of the quality of the assignment, that can be compared for multiple values of k .

Note that this approach protects us against the each-point-in-its-cluster situation: in this scenario, the distance between clusters would decrease really rapidly, meaning that the values in \mathbf{R} would *increase*; the Davies-Bouldin measure indicates a better clustering when the values are *lower*.

There is very little enumeration of techniques in this book. The important point is to understand how all of the pieces fit together, not to make a census of all possible pieces.

There are alternatives to this method, including silhouettes (Rousseeuw 1987) and the technique of Dunn (1974). The question of optimizing the number of clusters goes back several decades (Thorndike 1953), and it still actively studied. What matter is less to give a comprehensive overview of all the measures: the message here is to pick one that works (and can be justified) for your specific problem!

2.4. Application: optimal clustering of the satellite image data

2.4.1. Initial run

Before we do anything else, we need to run our algorithm with a random pick of hyper-parameters, in order to get a sense of how hard the task ahead is. In this case, using $k = 3$, we get the results presented in Figure 2.3. Why $k = 3$? Why not! The point of k -means is to provide a division of data in a small (manageable) number of classes, and the folklore around this method suggests that two is too few, and four is initially too much. Therefore, $k = 3$. We can tweak it later, and indeed we will, but what matters immediately is to generate a result to kickstart our learning process.

Take some time to think about how you would use k -means to come up with a way to remove pixels with only water from this image! We will actually perform this step in the next section.

It is always a good idea to look at the first results and state the obvious. Here, for example, we can say that water is easy to identify. In fact, removing open water pixels from images is an interesting image analysis challenge (Mondejar & Tongco 2019), and because we used an index that specifically identifies water bodies (NDWI), it is not surprising that there is an entire cluster that seems to be associated with water. But if we take a better look, it appears that there groups of pixels representing the area that burned that are classified with the water pixels. When looking at the landscape in a space with four dimensions, it looks like separating water from this type of landscape is difficult!

2.4. Application: optimal clustering of the satellite image data

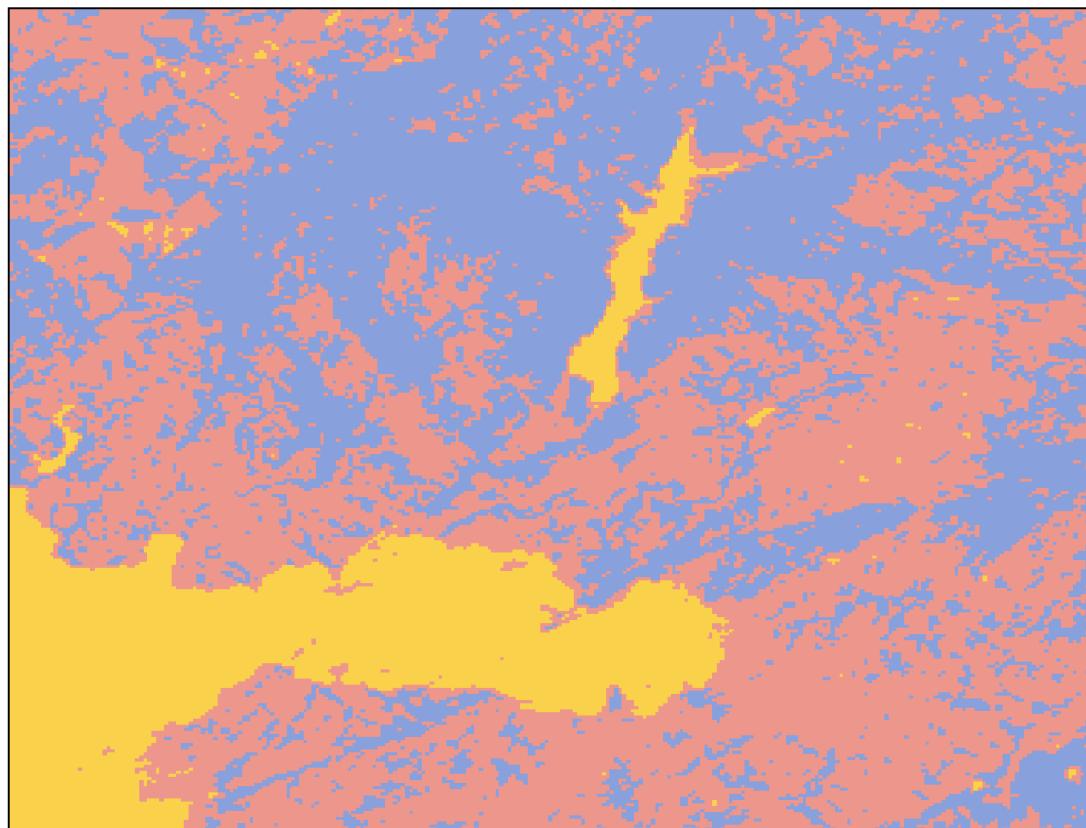


Figure 2.3.: After iterating the k-means algorithm, we obtain a classification for every pixel in the landscape. This classification is based on the values of NDVI, NDMI, and NDWI indices, and therefore groups pixels based on specific assumptions about vegetation and stress. This clustering was produced using $k = 3$, i.e. we want to see what the landscape would look like when divided into three categories.

2. Clustering

This might seem like an idle observation, but this is not the case! It means that when working on vegetation-related questions, we will likely need at least one cluster for water, and one cluster for burnt, and built-up areas. This is helpful information, because we can already think about how many classes of vegetation we are willing to accept, and add (at least) two clusters to capture other types of cover. Looking at the initial division in classes, it seems that having only two clusters for the variety of non-water pixels is not really sufficient, and so we start searching for an optimal division of the landscape.

2.4.2. Optimal number of pixels

We will revisit the issue of tuning the hyper-parameters in more depth in
?@sec-tuning

In order to produce Figure 2.3, we had to guess at a number of classes we wanted to split the landscape into. This introduces two important steps in coming up with a model: starting with initial parameters in order to iterate rapidly, and then refining these parameters to deliver a model that is fit for purpose. Our discussion in Section 2.4.1, where we concluded that we needed to keep (maybe) two classes for water and built-up is not really satisfying, as we do not yet have a benchmark to evaluate the correct value of k ; we know that it is more than 3, but how much more?

We will now change the values of k and use the Davies & Bouldin (1979) measure introduced in Section 2.3.5 to identify the optimal value of k . The results are presented in Figure 2.4. Note that we only explore $k \in [3, 10]$. More than 10 categories is probably not very actionable, and therefore we can make the decision to only look at this range of parameters. Sometimes (always!) the best solution is the one that gets your job done.

Note that we will play a little trick on our data here. We know from Figure 2.3 that with $k = 3$, we can do a good job at removing water pixels, which seem to cover about a third of the image. If we are interested in land-based questions, this is quite a lot of information we do not really care about. In order to make the clustering more relevant to our interests, we will therefore extract the pixels that are *not* classified as water before doing the clustering. This *will* miss a few relevant pixels, but for now we can consider this an acceptable trade-off in order to have a better classification of land pixels.

What would make such a classification better? Recall from the Section 2.3 that the assignment of instances to classes is done based on the distance to the centroid, which it itself the average of the features of the members of each class. By removing the pixels that are not relevant to our question (such as using water pixels to classify

2.4. Application: optimal clustering of the satellite image data

the image using data relevant to vegetation), we are taking steps to ensure that the space in which the centroids can exist is more rigorously defined for the question we are interested in.

There are two interesting things in Figure 2.4. First, note that for $k = \{3, 4, 5, 6\}$, there is almost no dispersal: all of the assignments have the exact same score, which is *unlikely* to happen except if the assignments are the same every time! This is a good sign, and, anecdotally, something that might suggest a really information separation of the points. Second, $k = 4$ has the lowest Davies-Bouldin index of all values we tried, and is therefore strongly suggestive of an optimal hyper-parameter.

Note that using $k = 3$ on the non-water pixels is *different* from using $k = 4$ on all pixels! This is an important distinction, as it accentuates the importance of data selection and preparation on the process of picking good hyper-parameters.

2.4.3. Clustering with optimal number of classes

The clustering of pixels using $k = 4$ is presented in Figure 2.5. Unsurprisingly, k -means separated the burnt area, as well as the more forested/green areas. Now is a good idea to start thinking about what is representative of these clusters: one is associated with very high NDWI value (these are the water pixels), and two classes have both high NDVI and high NDMI (suggesting different categories of vegetation).

The relative size of the clusters (as well as the position of their centroids) is presented in Table 2.2. There is a good difference in the size of the clusters, which is an important thing to note. Indeed, a common myth about k -means is that it gives clusters of the same size. This “size” does not refer to the cardinality of the clusters, but to the volume that they cover in the space of the parameters. If an area of the space of parameters is more densely packed with instances, the cluster covering the area will have more points!

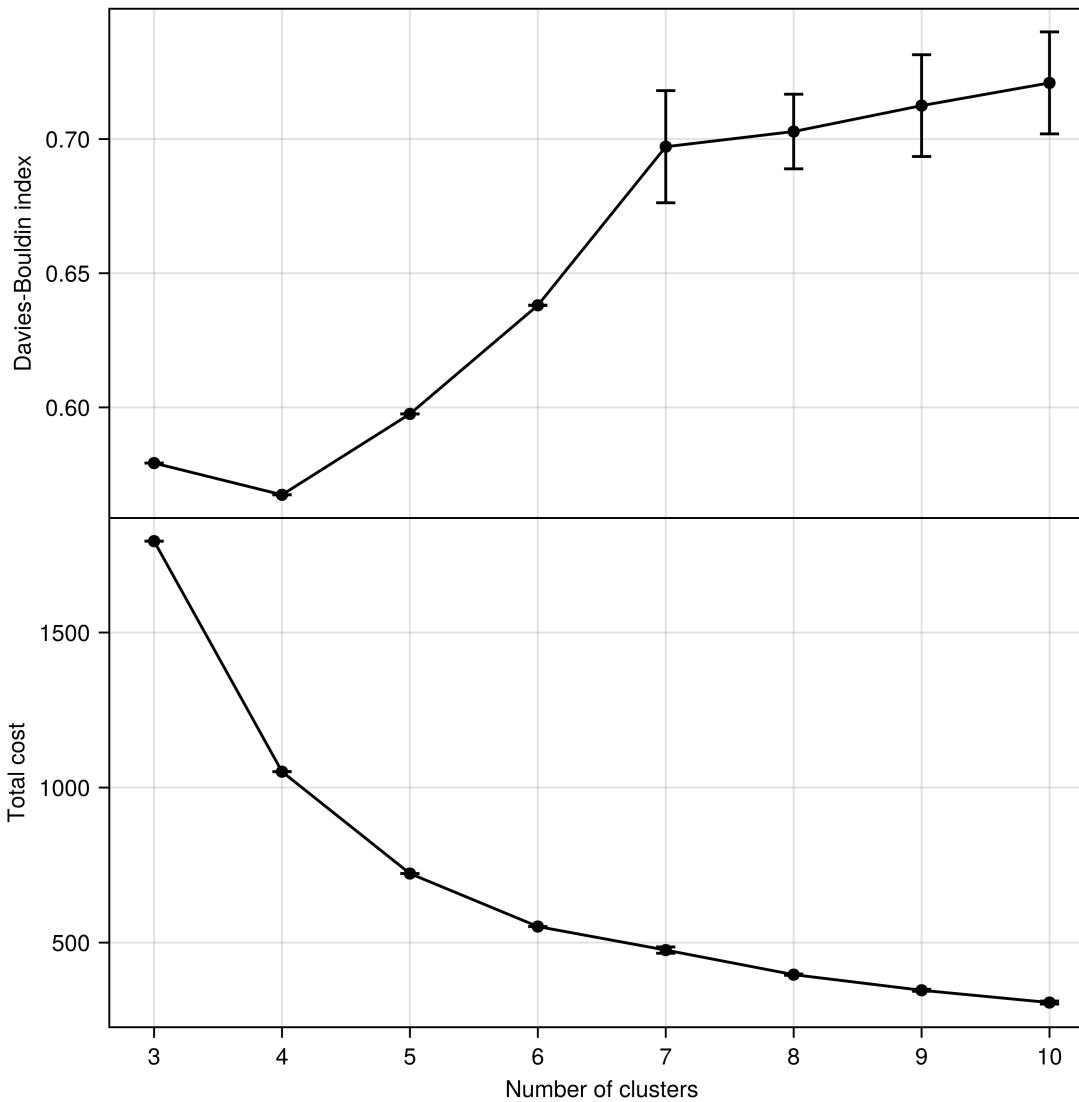
We will revisit the issue of understanding how a model makes a prediction in [?@sec-explanations](#).

Table 2.2.: Summary of the values for the centers of the optimal clusters found in this image. The cover column gives the percentage of all pixels associated to this class. The clusters are sorted by the NDVI of their centroid.

Cluster	Cover	NDVI	NDWI	NDMI	NBR
2	6	0.077	-0.1	-0.11	-0.102
3	26	0.197	-0.217	-0.025	0.085
4	35	0.261	-0.258	0.058	0.175
1	26	0.331	-0.309	0.145	0.269

2. Clustering

Figure 2.4.: Results of running the k -means algorithm ten times for each number of clusters between 3 and 10. The average Davies-Bouldin and cost are reported, as well as the standard deviation. As expected, the total cost decreases with more clusters, but this is not necessarily the sign of a better clustering.



2.4. Application: optimal clustering of the satellite image data

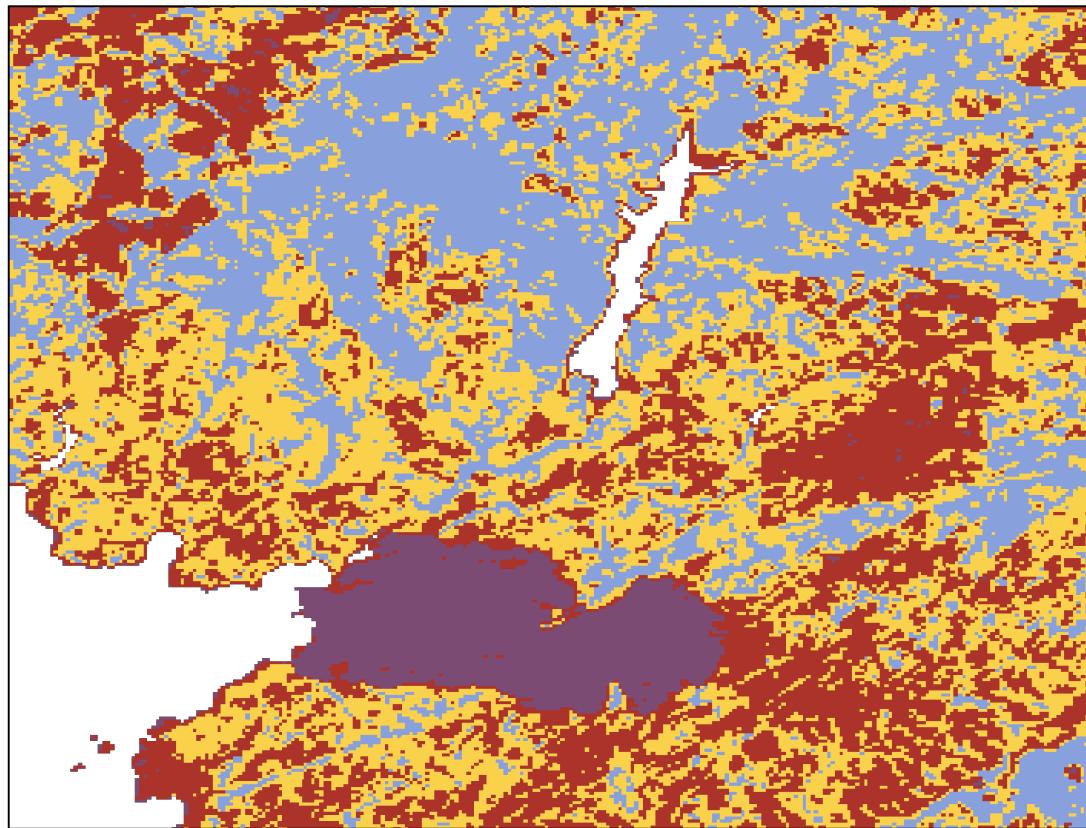


Figure 2.5.: Results of the landscape clustering with $k=4$ clusters. This number of clusters gives us a good separation between different groups of pixels, and seems to capture features of the landscape as revealed with the false-color composites.

2. Clustering

The area of the space of parameters covered by each cluster is represented in Figure 2.6, and this result is actually not surprising, if we spend some time thinking about how k -means work. Because our criteria to assign a point to a cluster is based on the being closest to its centroid than to any other centroid, we are essentially creating Voronoi cells, with linear boundaries between them.

This behavior makes k -means excellent at creating color palettes from images! Cases in point, [Karthik Ram's Wes Anderson palettes](#), and [David Lawrence Miller's Beyoncé palettes](#). Let it never again be said that ecologists should not be trusted with machine learning techniques.

By opposition to a model based on, for example, mixtures of Gaussians, the assignment of a point to a cluster in k -means is independent of the current composition of the cluster (modulo the fact that the current composition of the cluster is used to update the centroids). In fact, this makes k -means closer to (or at least most efficient as) a method for quantization (Gray 1984).

2.5. Is k -means a model?

Yes.

Well, it depends.

Deciding what we mean by “a model” is actually an interesting question to ponder. When we apply k -means to the data, it brings us additional information, specifically the class to which we can assign the different pixels. This is creating novel information from the data, and this should be sufficient to accept that k -means is a model.

But maybe we espouse a more ambitious definition of a model, and we are specifically looking for a model that can make prediction on data that we had not previously used. In that case, k -means still qualifies. Imagine, for a moment, a new pixel. This is a little unrealistic as, unless some unforeseen geological events took place, all of the pixels we could describe today for this dataset already existed in 2017. But imagine we changed the bounding box of the scene: we now have spectral data for a few thousand pixels we had not previously observed.

We do not need to run k -means again to classify these pixels. Because k -means is trained when we have established the number and position of the centroids, we can now simply figure out which cluster the new point belongs to, which is done using Equation 2.1.

2.5. Is k-means a model?

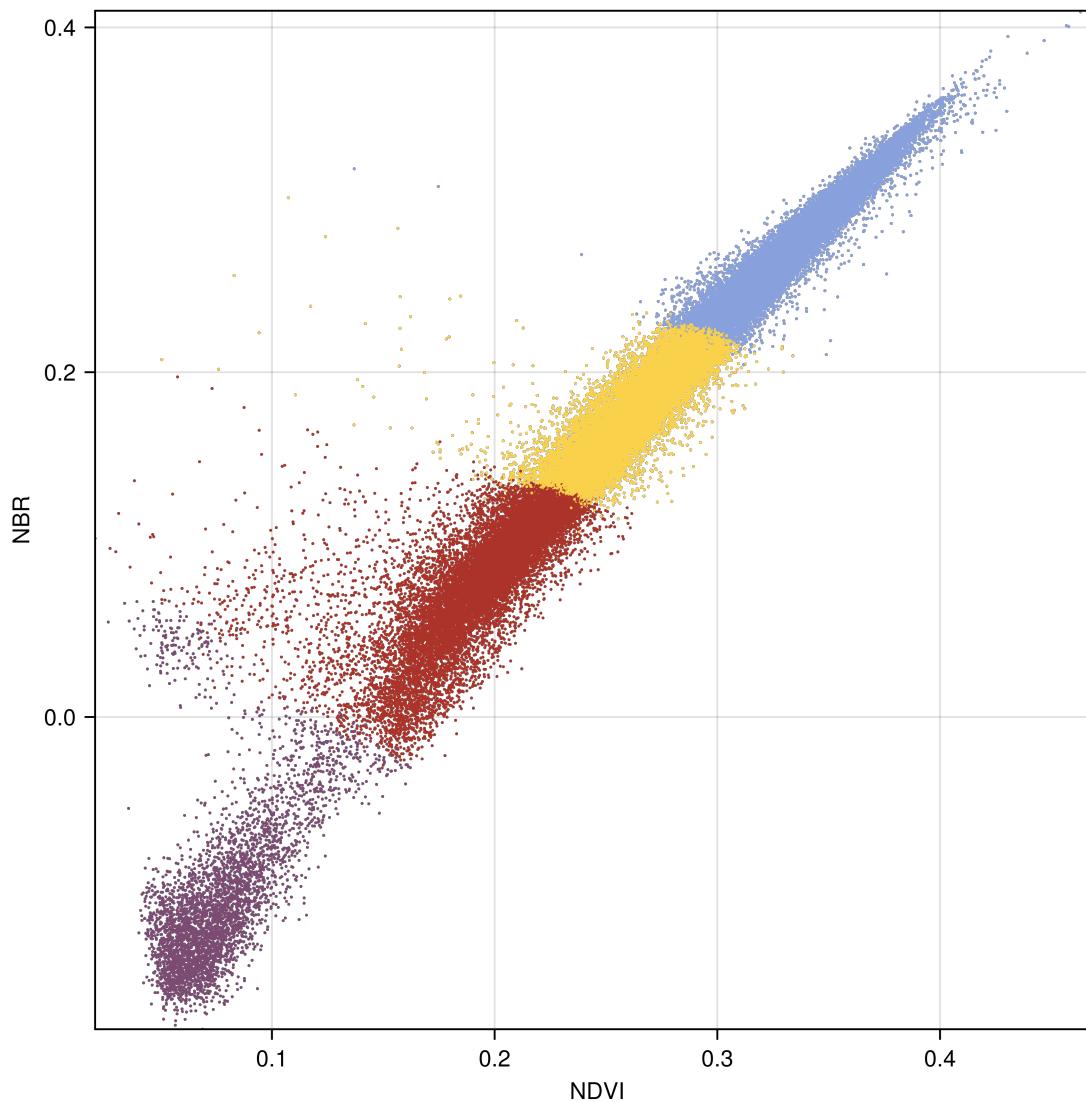


Figure 2.6.: Visualisation of the clustering output as a function of the NDVI and NBR values. Note that the limits between the clusters are lines (planes), and that each cluster covers about the same volume in the space of parameters.

2. Clustering

Both “what is a model” and “what is a new datapoint” are important questions to think about, and there are often edge cases to the answers we can bring. They will come back often in this book, notably in Chapter 4. It is always a good idea, at the end of a chapter, to stop and think for a moment about what our model *really* is. What problems can it can solve? What would it mean to use this model on new data? In doing so, you will naturally start thinking about the place of each model in the broader problem of “how do I make my data make sense?”.

2.6. Conclusion

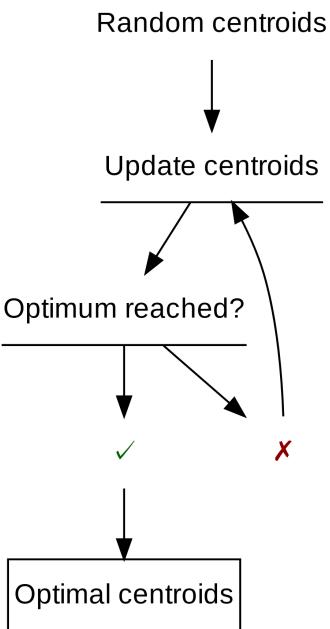
In this chapter, we have used the k -means algorithm to create groups in a large dataset that had no labels, *i.e.* the points were not initially assigned to a class. By picking the features we wanted to cluster the different points, we were able to highlight specific aspects of the landscape. In Chapter 3, we will start adding labels to our data, and shift our attention from classification to regression problems.

References

- Aloise, D., Deshpande, A., Hansen, P. & Popat, P. (2009). [NP-hardness of Euclidean sum-of-squares clustering](#). *Machine Learning*, 75, 245–248.
- Cooney, C.R., He, Y., Varley, Z.K., Nouri, L.O., Moody, C.J.A., Jardine, M.D., *et al.* (2022). [Latitudinal gradients in avian colourfulness](#). *Nature Ecology & Evolution*, 6, 622–629.
- Cooper, N., Bond, A.L., Davis, J.L., Portela Miguez, R., Tomsett, L. & Helgen, K.M. (2019). [Sex biases in bird and mammal natural history collections](#). *Proceedings of the Royal Society B: Biological Sciences*, 286, 20192025.
- Dansereau, G., Barros, C. & Poisot, T. (2024). [Spatially explicit predictions of food web structure from regional-level data](#). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 379.
- Davies, D.L. & Bouldin, D.W. (1979). [A cluster separation measure](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1, 224–227.
- Dove, S., Böhm, M., Freeman, R., Jellesmark, S. & Murrell, D.J. (2023). [A user-friendly guide to using distance measures to compare time series in ecology](#). *Ecology and Evolution*, 13.

- Dunn, J.C. (1974). Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 4, 95–104.
- Gonzalez, A., Vihervaara, P., Balvanera, P., Bates, A.E., Bayraktarov, E., Bellingham, P.J., et al. (2023). A global biodiversity observing system to unite monitoring and guide action. *Nature Ecology & Evolution*.
- Gray, R. (1984). Vector quantization. *IEEE ASSP Magazine*, 1, 4–29.
- Kennedy, S. & Burbach, M. (2020). Great Plains Ranchers Managing for Vegetation Heterogeneity: A Multiple Case Study. *Great Plains Research*, 30, 137–148.
- Legendre, P. & Gauthier, O. (2014). Statistical methods for temporal and space–time analysis of community composition data. *Proceedings of the Royal Society B: Biological Sciences*, 281, 20132728.
- Luccioni, A.S. & Rolnick, D. (2023). Bugs in the data: How ImageNet misrepresents biodiversity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 14382–14390.
- McFeeters, S. (2013). Using the Normalized Difference Water Index (NDWI) within a Geographic Information System to Detect Swimming Pools for Mosquito Abatement: A Practical Approach. *Remote Sensing*, 5, 3544–3561.
- Mondejar, J.P. & Tongco, A.F. (2019). Near infrared band of Landsat 8 as water index: a case study around Cordova and Lapu-Lapu City, Cebu, Philippines. *Sustainable Environment Research*, 29.
- Rousseeuw, P.J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.
- Roy, D.P., Boschetti, L. & Trigg, S.N. (2006). Remote Sensing of Fire Severity: Assessing the Performance of the Normalized Burn Ratio. *IEEE Geoscience and Remote Sensing Letters*, 3, 112–116.
- Thorndike, R.L. (1953). Who belongs in the family? *Psychometrika*, 18, 267–276.
- Vermote, E., Justice, C., Claverie, M. & Franch, B. (2016). Preliminary analysis of the performance of the Landsat 8/OLI land surface reflectance product. *Remote Sensing of Environment*, 185, 46–56.
- Zheng, Y., Zhou, Q., He, Y., Wang, C., Wang, X. & Wang, H. (2021). An Optimized Approach for Extracting Urban Land Based on Log-Transformed DMSP-OLS Nighttime Light, NDVI, and NDWI. *Remote Sensing*, 13, 766.

2. Clustering



Flowchart 2.1: An overview of the process of coming up with a usable model. The process of creating a model starts with a training 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 then applied to an independent testing dataset, and the evaluation of the performance determines whether it will be used.

3. Gradient descent

As we progress into this book, the process of delivering a trained model is going to become more and more complex. In Chapter 2, we worked with a model that did not really require training (but did require to pick the best hyper-parameter). In this chapter, we will only increase complexity very slightly, by considering how we can train a model when we have a reference dataset to compare to.

Doing so will require to introduce several new concepts, and so the “correct” way to read this chapter is to focus on the high-level process. The problem we will try to solve (which is introduced in Section 3.2) is very simple; in fact, the empirical data looks more fake than many simulated datasets!

3.1. A digression: what is a trained model?

Models are data. When a model is trained, it represents a series of measurements (its parameters), taken on a representation of the natural world (the training data), through a specific instrument (the model itself, see *e.g.* Morrison & Morgan 1999). A trained model is, therefore, capturing our understanding of a specific situation we encountered. We need to be very precise when defining what, exactly, a model describes. In fact, we need to take a step back and try to figure out where the model stops.

As we will see in this chapter, then in Chapter 4, and finally in `?@sec-tuning`, the fact of training a model means that there is a back and forth between the algorithm we train, the data we use for training, and the criteria we set to define the performance of the trained model. The algorithm bound to its dataset is the *machine* we train in machine learning.

3. Gradient descent

Therefore, a trained model is never independent from its training data: they describe the scope of the problem we want to address with this model. In Chapter 2, we ended up with a machine (the trained k -means algorithm) whose parameters (the centroids of the classes) made sense in the specific context of the training data we used; applied to a different dataset, there are no guarantees that our model would deliver useful information.

For the purpose of this book, we will consider that a model is trained when we have defined the algorithm, the data, the measure through which we will evaluate the model performance, and then measured the performance on a dataset built specifically for this task. All of these elements are important, as they give us the possibility to *explain* how we came up with the model, and therefore, how we made the predictions. This is different from reasoning about why the model is making a specific prediction (we will discuss this in [?@sec-explanations](#)), and is more related to explaining the process, the “outer core” of the model. As you read this chapter, pay attention to these elements: what algorithm are we using, on what data, how do we measure its performance, and how well does it perform?

3.2. The problem: how many interactions in a food web?

One of the earliest observation that ecologists made about food webs is that when there are more species, there are more interactions. A remarkably insightful crowd, food web ecologists. Nevertheless, it turns out that this apparently simple question had received a few different answers over the years.

The initial model was proposed by Cohen & Briand (1984): the number of interactions L scales linearly with the number of species S . After all, we can assume that when averaging over many consumers, there will be an average diversity of resources they consume, and so the number of interactions could be expressed as $L \approx b \times S$.

Not so fast, said Martinez (1992). When we start looking at food webs with more species, the increase of L with regards to S is superlinear. Thinking in ecological terms, maybe we can argue that consumers are flexible, and that instead of sampling a set number of resources, they will sample a set proportion of the number of consumer-resource combinations (of which there are S^2). In this interpretation, $L \approx b \times S^2$.

But the square term can be relaxed; and there is no reason not to assume a power law, with $L \approx b \times S^a$. This last formulation has long been accepted as the most workable one, because it is possible to approximate values of its parameters using other ecological processes (Brose *et al.* 2004).

The “reality” (*i.e.* the relationship between S and L that correctly accounts for ecological constraints, and fit the data as closely as possible) is a little bit different than this formula (MacDonald *et al.* 2020). But for the purpose of this chapter, figuring out the values of a and b from empirical data is a very instructive exercise.

In Figure 3.1, we can check that there is a linear relationship between the natural log of the number of species and the natural log of the number of links. This is not surprising! If we assume that $L \approx b \times S^a$, then we can take the log of both sides, and we get $\log L \approx a \times \log S + \log b$. This is linear model, and so we can estimate its parameters using linear regression!

3.3. Gradient descent

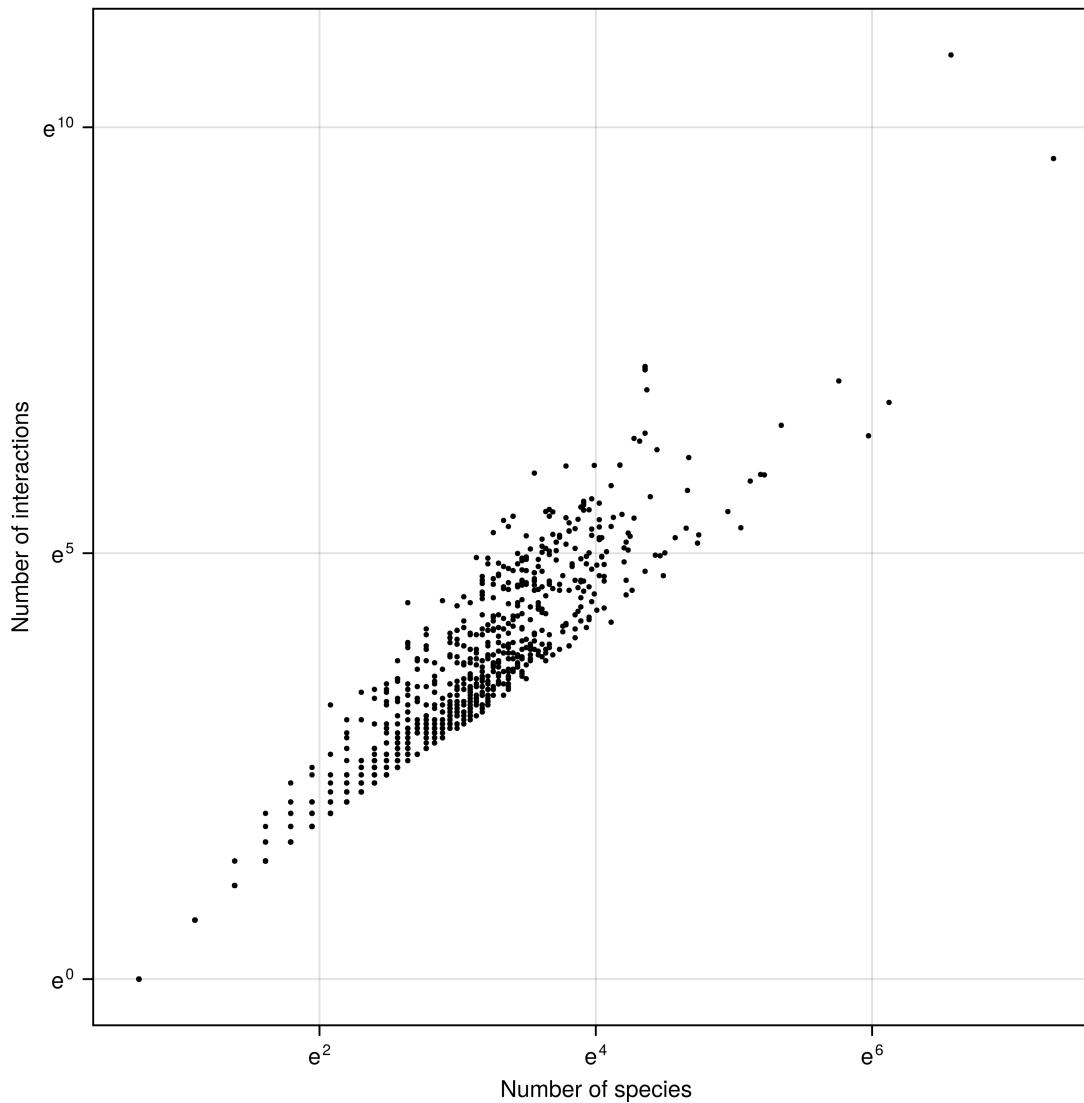
Gradient descent is built around a remarkably simple intuition: knowing the formula that gives rise to our prediction, and the value of the error we made for each point, we can take the derivative of the error with regards to each parameter, and this tells us how much this parameter contributed to the error. Because we are taking the derivative, we can further know whether to increase, or decrease, the value of the parameter in order to make a smaller error next time.

In this section, we will use linear regression as an example, because it is the model we have decided to use when exploring our ecological problem in Section 3.2, and because it is suitably simple to keep track of everything when writing down the gradient by hand.

Before we start assembling the different pieces, we need to decide what our model is. We have settled on a linear model, which will have the form $\hat{y} = m \times x + b$. The little hat on \hat{y} indicates that this is a prediction. The input of this model is x , and its parameters are m (the slope) and b (the intercept). Using the notation we adopted in Section 3.2, this would be $\hat{l} = a \times s + b$, with $l = \log L$ and $s = \log S$.

3. Gradient descent

Figure 3.1.: We have assumed that the relationship between L and S could be represented by $L \approx b \times S^a$, which gave us a reason to take the natural log of both variables. On this figure, we see that the relationship between the logs look linear, which means that linear regression has a good chance of estimating the values of the parameters.



3.3.1. Defining the loss function

The loss function is an important concept for anyone attempting to compare predictions to outcomes: it quantifies how far away an ensemble of predictions is from a benchmark of known cases. There are many loss functions we can use, and we will indeed use a few different ones in this book. But for now, we will start with a very general understanding of what these functions *do*.

Think of prediction as throwing a series of ten darts on ten different boards. In this case, we know what the correct outcome is (the center of the board, I assume, although I can be mistaken since I have only played darts once, and lost). A cost function would be any mathematical function that compares the position of each dart on each board, the position of the correct event, and returns a score that informs us about how poorly our prediction lines up with the reality.

In the above example, you may be tempted to say that we can take the Euclidean distance of each dart to the center of each board, in order to know, for each point, how far away we landed. Because there are several boards, and because we may want to vary the number of boards while still retaining the ability to compare our performances, we would then take the average of these measures.

We will note the position of our dart as being \hat{y} , the position of the center as being y (we will call this the *ground truth*), and the number of attempts n , and so we can write our loss function as

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.1)$$

This loss function is usually called the MSE (Mean Standard Error), or L2 loss, or the quadratic loss, because the paths to machine learning terminology are many. This is a good example of a loss function for regression (and we will discuss loss functions for classification later in this book). There are alternative loss functions to use for regression problems in Table 3.1.

In data science, things often have multiple names. This is true of loss functions, and this will be even more true on other things later.

3. Gradient descent

Table 3.1.: List of common loss functions for regression problems

Measure	Expression	Remarks
Mean Squared Error (MSE, L2)	$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$	Large errors are (proportionally) more penalized because of the squaring
Mean Absolute Error (MAE, L1)	$\frac{1}{n} \sum_{i=1}^n \ y_i - \hat{y}_i\ $	Error measured in the units of the response variable
Root Mean Square Error (RMSE)	$\sqrt{\text{MSE}}$	Error measured in the units of the response variable
Mean Bias Error	$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)$	Errors <i>can</i> cancel out, but this can be used as a measure of positive/negative bias

Throughout this chapter, we will use the L2 loss (Equation 3.1), because it has *really* nice properties when it comes to taking derivatives, which we will do a lot of. In the case of a linear model, we can rewrite Equation 3.1 as

$$f = \frac{1}{n} \sum (y_i - m \times x_i - b)^2 \quad (3.2)$$

There is an important change in Equation 3.2: we have replaced the prediction \hat{y}_i with a term that is a function of the predictor x_i and the model parameters: this means that we can calculate the value of the loss as a function of a pair of values (x_i, y_i) , and the model parameters.

3.3.2. Calculating the gradient

With the loss function corresponding to our problem in hands (Equation 3.2), we can calculate the gradient. Given a function that is scalar-valued (it returns a single value), taking several variables, that is differentiable, the gradient of this function is a vector-valued (it returns a vector) function; when evaluated at a specific point,

3.3. Gradient descent

this vectors indicates both the direction and the rate of fastest increase, which is to say the direction in which the function increases away from the point, and how fast it moves.

We can re-state this definition using the terms of the problem we want to solve. At a point $p = [m \ b]^\top$, the gradient ∇f of f is given by:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial m}(p) \\ \frac{\partial f}{\partial b}(p) \end{bmatrix}. \quad (3.3)$$

This indicates how changes in m and b will *increase* the error. In order to have a more explicit formulation, all we have to do is figure out an expression for both of the partial derivatives. In practice, we can let auto-differentiation software calculate the gradient for us (Innes 2018); these packages are now advanced enough that they can take the gradient of code directly.

Solving $(\partial f / \partial m)(p)$ and $(\partial f / \partial c)(p)$ is easy enough:

$$\nabla f(p) = \begin{bmatrix} -\frac{2}{n} \sum [x_i \times (y_i - m \times x_i - b)] \\ -\frac{2}{n} \sum (y_i - m \times x_i - b) \end{bmatrix}. \quad (3.4)$$

Note that both of these partial derivatives have a term in $2n^{-1}$. Getting rid of the 2 in front is very straightforward! We can modify Equation 3.2 to divide by $2n$ instead of n . This modified loss function retains the important characteristics: it increases when the prediction gets worse, and it allows comparing the loss with different numbers of points. As with many steps in the model training process, it is important to think about *why* we are doing certain things, as this can enable us to make some slight changes to facilitate the analysis.

With the gradient written down in Equation 3.4, we can now think about what it means to *descend* the gradient.

3. Gradient descent

3.3.3. Descending the gradient

Recall from Section 3.3.2 that the gradient measures how far we *increase* the function of which we are taking the gradient. Therefore, it measures how much each parameter contributes to the loss value. Our working definition for a trained model is “one that has little loss”, and so in an ideal world, we could find a point p for which the gradient is as small as feasible.

Because the gradient measures how far away we increase error, and intuitive way to use it is to take steps in the *opposite* direction. In other words, we can update the value of our parameters using $p := p - \nabla f(p)$, meaning that we subtract from the parameter values their contribution to the overall error in the predictions.

But, as we will discuss further in Section 3.3.4, there is such a thing as “too much learning”. For this reason, we will usually not move the entire way, and introduce a term to regulate how much of the way we actually want to descend the gradient. Our actual scheme to update the parameters is

$$p := p - \eta \times \nabla f(p). \quad (3.5)$$

This formula can be *iterated*: with each successive iteration, it will get us closer to the optimal value of p , which is to say the combination of m and b that minimizes the loss.

3.3.4. A note on the learning rate

The error we can make on the first iteration will depend on the value of our initial pick of parameters. If we are *way off*, especially if we did not re-scale our predictors and responses, this error can get very large. And if we make a very large error, we will have a very large gradient, and we will end up making very big steps when we update the parameter values. There is a real risk to end up over-compensating, and correcting the parameters too much.

In order to protect against this, in reality, we update the gradient only a little, where the value of “a little” is determined by an hyper-parameter called the *learning rate*, which we noted η . This value will be very small (much less than one). Picking the correct learning rate is not simply a way to ensure that we get correct results

3.4. Application: how many links are in a food web?

(though that is always a nice bonus), but can be a way to ensure that we get results *at all*. The representation of numbers in a computer’s memory is tricky, and it is possible to create an overflow: a number so large it does not fit within 64 (or 32, or 16, or however many we are using) bits of memory.

The conservative solution of using the smallest possible learning rate is not really effective, either. If we almost do not update our parameters at every epoch, then we will take almost forever to converge on the correct parameters. Figuring out the learning rate is an example of hyper-parameter tuning, which we will get back to later in this book.

3.4. Application: how many links are in a food web?

We will not get back to the problem exposed in Figure 3.1, and use gradient descent to fit the parameters of the model defined as $\hat{y} \approx \beta_0 + \beta_1 \times x$, where, using the notation introduced in Section 3.2, \hat{y} is the natural log of the number of interactions (what we want to predict), x is the natural log of the species richness (our predictor), and β_0 and β_1 are the parameters of the model.

3.4.1. The things we won’t do

At this point, we could decide that it is a good idea to transform our predictor and our response, for example using the z-score. But this is not really required here; we know that our model will give results that make sense in the units of species and interactions (after dealing with the natural log, of course). In addition, as we will see in [?@sec-leakage](#), applying a transformation to the data too soon can be a dangerous thing. We will have to live with raw features for a few more chapters.

In order to get a sense of the performance of our model, we will remove some of the data, meaning that the model will not learn on these data points. We will get back to this practice (cross-validation) in a lot more details in Chapter 4, but for now it is enough to say that we hide 20% of the dataset, and we will use them to evaluate how good the model is as it trains. The point of this chapter is not to think too deeply about cross-validation, but simply to develop intuitions about the way a machine learns.

3. Gradient descent

3.4.2. Starting the learning process

In order to start the gradient descent process, we need to decide on an initial value of the parameters. There are many ways to do it. We could work our way from our knowledge of the system; for example $b < 1$ and $a = 2$ would fit relatively well with early results in the food web literature. Or we could draw a pair of values (a, b) at random. Looking at Figure 3.1, it is clear that our problem is remarkably simple, and so presumably either solution would work.

3.4.3. Stopping the learning process

The gradient descent algorithm is entirely contained in Equation 3.5 , and so we only need to iterate several times to optimize the parameters. How long we need to run the algorithm for depends on a variety of factors, including our learning rate (slow learning requires more time!), our constraints in terms of computing time, but also how good we need to model to be.

The number of iterations over which we train the model is usually called the number of epochs, and is an hyper-parameter of the model.

One usual approach is to decide on a number of iterations (we need to start somewhere), and to check how rapidly the model seems to settle on a series of parameters. But more than this, we also need to ensure that our model is not learning *too much* from the data. This would result in over-fitting, in which the models gets better on the data we used to train it, and worse on the data we kept hidden from the training! In Table 3.2, we present the RMSE loss for the training and testing datasets, as well as the current estimates of the values of the parameters of the linear model.

Table 3.2.: This table shows the change in the model, as measured by the loss and by the estimates of the parameters, after an increasing amount of training epochs. The loss drops sharply in the first 500 iterations, but even after 20000 iterations, there are still some changes in the values of the parameters.

Step	Loss (training)	Loss (testing)	β_0	β_1
1	3.78688	3.74325	0.4	0.2
10	2.8993	2.87734	0.485721	0.22631
30	1.67104	1.67734	0.635904	0.270871
100	0.511208	0.537535	0.899103	0.33642

3.4. Application: how many links are in a food web?

Step	Loss (training)	Loss (testing)	β_0	β_1
300	0.369604	0.393303	1.02996	0.311367
1000	0.306995	0.330739	1.10205	0.115279
3000	0.21051	0.23233	1.24419	-0.296017
10000	0.152797	0.16681	1.43175	-0.838781
20000	0.150743	0.16238	1.47075	-0.951649

In order to protect against over-fitting, it is common to add a check to the training loop, to say that after a minimum number of iterations has been done, we stop the training when the loss on the testing data starts increasing. In order to protect against very long training steps, it is also common to set a tolerance (absolute or relative) under which we decide that improvements to the loss are not meaningful, and which serves as a stopping criterion for the training.

3.4.4. Detecting over-fitting

As we mentioned in the previous section, one risk with training that runs for too long is to start seeing over-fitting. The usual diagnosis for over-fitting is an increase in the testing loss, which is to say, in the loss measured on the data that were not used for training. In Figure 3.2, we can see that the RMSE loss decreases at the same rate on both datasets, which indicates that the model is learning from the data, but not to a point where its ability to generalize suffers.

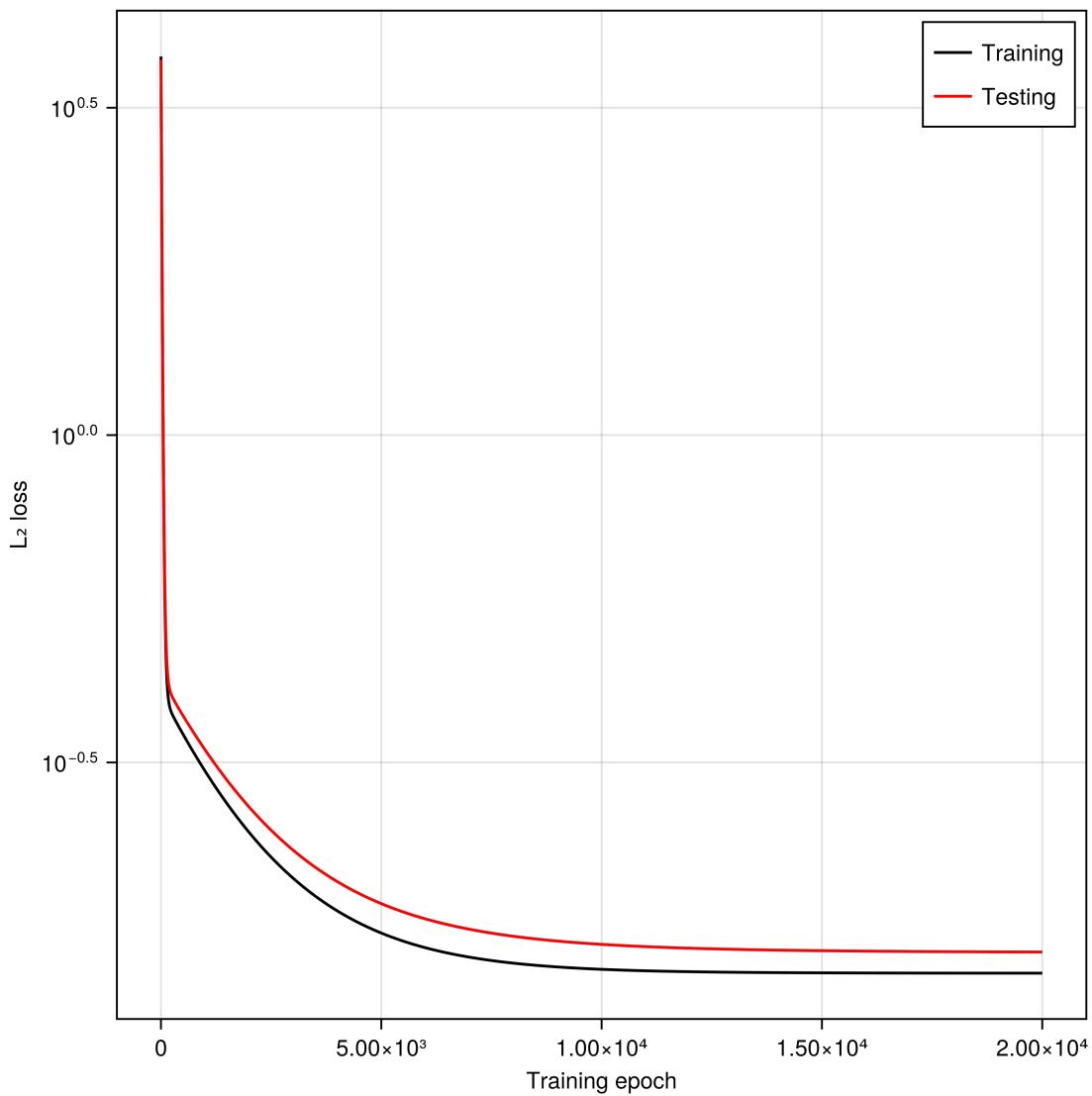
We are producing the loss over time figure after the training, as it is good practice – but as we mentioned in the previous section, it is very common to have the training code look at the dynamics of these two values in order to decide whether to stop the training early.

Before moving forward, let's look at Figure 3.2 a little more closely. In the first steps, the loss decreases very rapidly – this is because we started from a value of β that is, presumably, far away from the optimum, and therefore the gradient is really strong. Despite the low learning rate, we are making long steps in the space of parameters. After this initial rapid increase, the loss decreases much more slowly. This, counter-intuitively, indicates that we are getting closer to the optimum! At the exact point where β_0 and β_1 optimally describe

Underfitting is also a possible scenario, where the model is *not* learning from the data, and can be detected by seeing the loss measures remain high or even increase.

3. Gradient descent

Figure 3.2.: This figure shows the change in the loss for the training and testing dataset. As the two curves converge on low values at the same rate, this suggests that the model is not over-fitting, and is therefore suitable for use.



3.4. Application: how many links are in a food web?

our dataset, the gradient vanishes, and our system would stop moving. And as we get closer and closer to this point, we are slowing down. In the next section, we will see how the change in loss over times ties into the changes with the optimal parameter values.

3.4.5. Visualizing the learning process

From Figure 3.3, we can see the change in β_0 and β_1 , as well as the movement of the current best estimate of the parameters (right panel). The sharp decrease in loss early in the training is specifically associated to a rapid change in the value of β_0 . Further note that the change in parameters values is *not* monotonous! The value of β_1 initially increases, but when β_0 gets closer to the optimum, the gradient indicates that we have been moving β_1 in the “wrong” direction.

This is what gives rise to the “elbow” shape in the right panel of Figure 3.3. Remember that the gradient descent algorithm, in its simple formulation, assumes that we can *never* climb back up, *i.e.* we never accept a costly move. The trajectory of the parameters therefore represents the path that brings them to the lowest point they can reach *without* having to temporarily recommend a worse solution.

But how good is the solution we have reached?

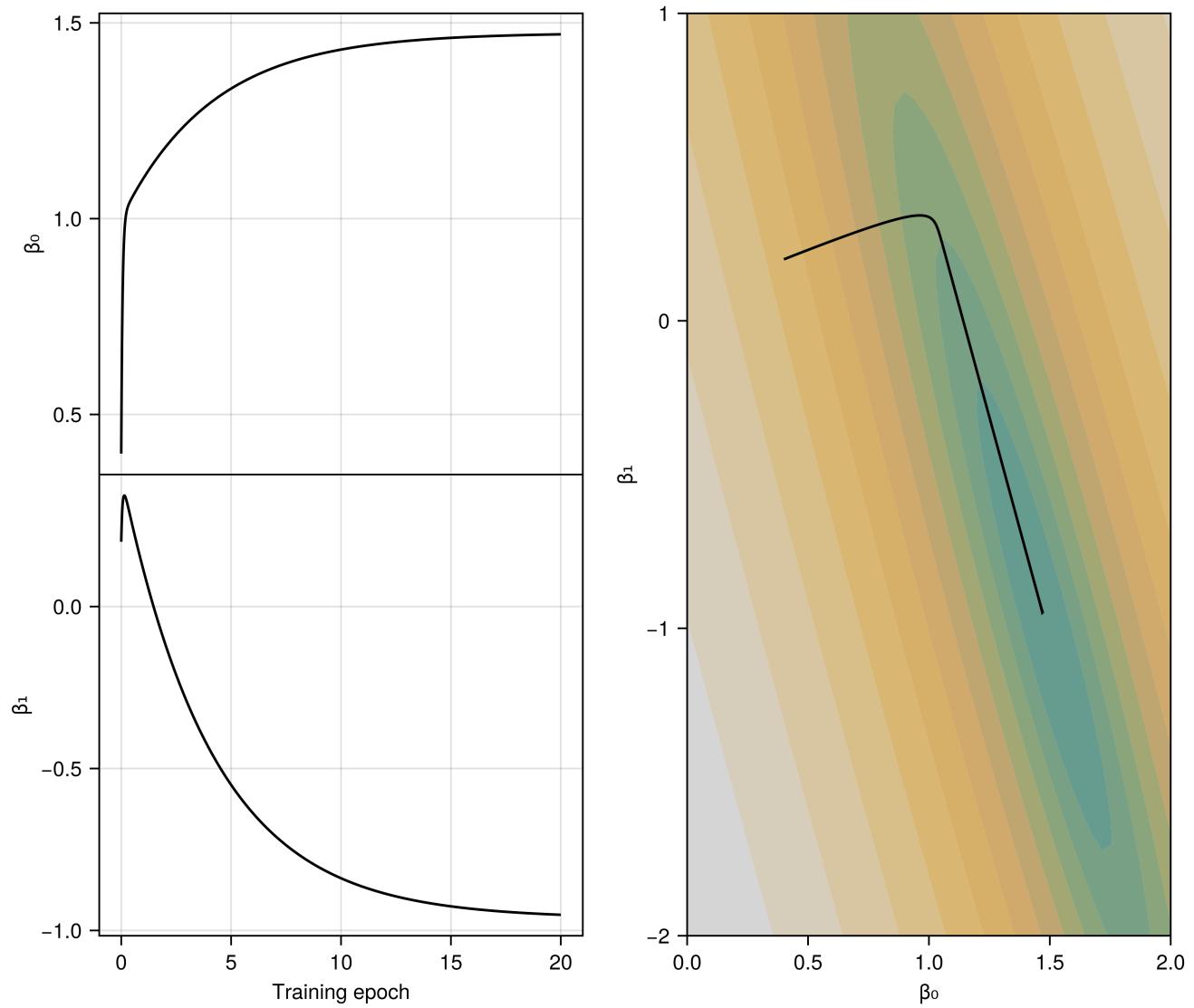
3.4.6. Outcome of the model

We could read the performance of the model using the data in Figure 3.2, but what we *really* care about is the model’s ability to tell us something about the data we initially gave it. This is presented in Figure 3.4. As we can see, the model is doing a rather good job at capturing the relationship between the number of species and the number of interactions.

We will have a far more nuanced discussion of “what is this model good for?” in Chapter 4, but for now, we can make a decision about this model: it provides a good approximation of the relationship between the species richness, and the number of interactions, in a food web.

3. Gradient descent

Figure 3.3.: This figure shows the change in the parameters values over time. Note that the change is very large initially, because we make large steps when the gradient is strong. The rate of change gets much lower as we get nearer to the “correct” value.



3.4. Application: how many links are in a food web?

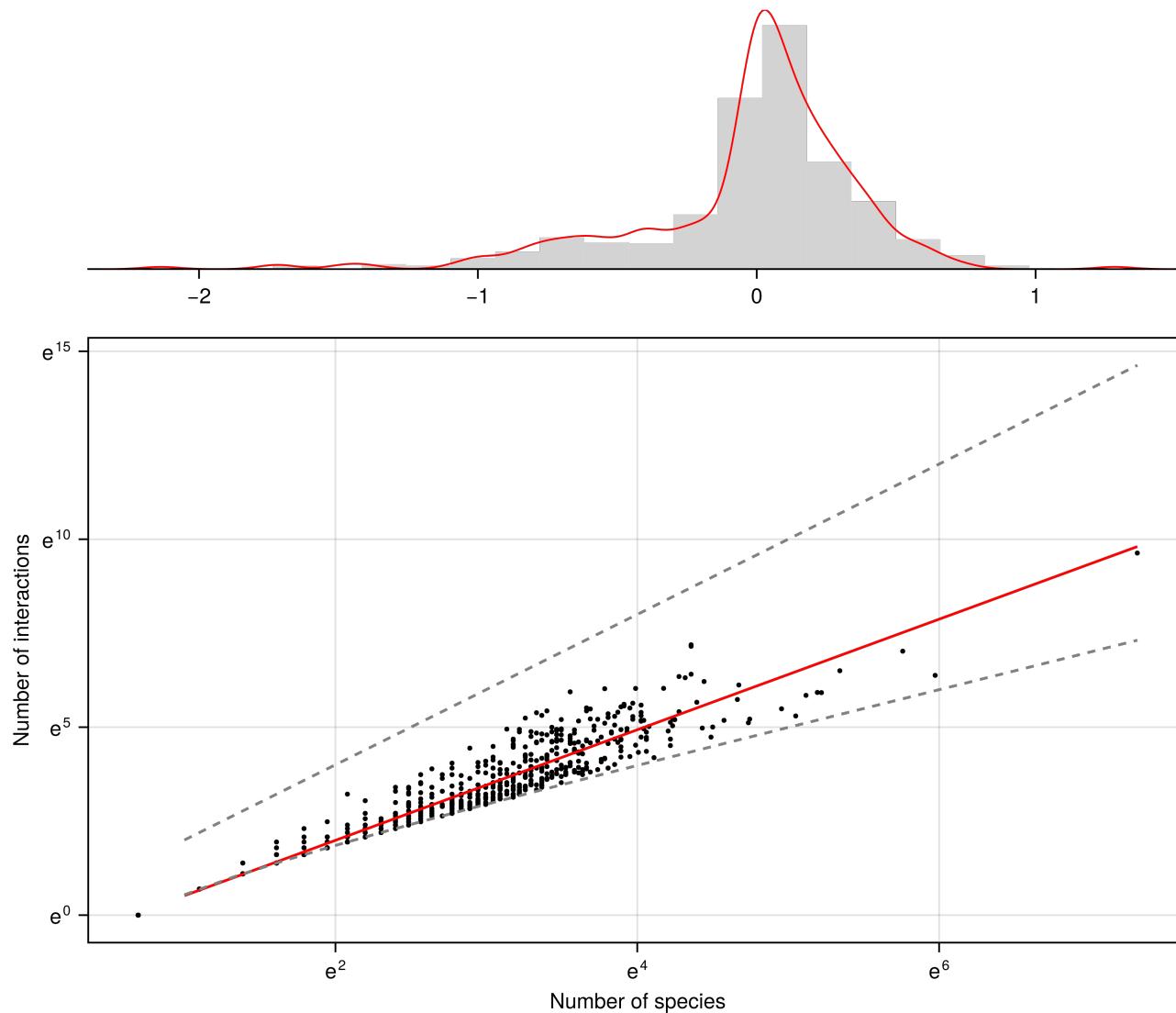


Figure 3.4.: Overview of the fitted model. The residuals (top panel) are mostly centered around 0, which suggests little bias towards over/under predicting interactions. The red line (based on the optimal coefficients) goes through the points, and indicates a rather good fit of the model.

3. Gradient descent

3.5. A note on regularization

One delicate issue that we have avoided in this chapter is the absolute value of the parameters. In other words, we didn't really care about how large the model parameters would be, only the quality of the fit. This is (generally) safe to do in a model with a single parameter. But what if we had many different terms? What if, for example, we had a linear model of the form $\hat{y} \approx \beta_0 + \beta_1 x + \beta_2 x^2$? What if our model was of the form $\hat{y} \approx \beta_0 + \beta_1 x + \dots + \beta_n x^n$? What if n started to get very large compared to the number of data points?

In this situation, we would very likely see overfitting, wherein the model would use the polynomial terms we provided to capture more and more noise in the data. This would be a dangerous situation, as the model will lose its ability to work on unknown data!

To prevent this situation, we may need to use regularization. Thanfkully, regularization is a relatively simple process. In Equation 3.4, the function $f(p)$ we used to measure the gradient was the loss function directly. In regularization, we use a slight variation on this, where

$$f(p) = \text{loss} + \lambda \times g(\beta),$$

where λ is an hyper-parameter giving the strength of the regularization, and $g(\beta)$ is a function to calculate the total penalty of a set of parameters.

When using $L1$ regularization (LASSO regression), $g(\beta) = \sum |\beta|$, and when using $L2$ regularization (ridge regression), $g(\beta) = \sum \beta^2$. When this gets larger, which happens when the absolute value of the parameters increases, the model is penalized. Note that if $\lambda = 0$, we are back to the initial formulation of the gradient, where the parameters have no direct effect on the cost.

3.6. Conclusion

In this chapter, we have used a dataset of species richness and number of interactions to start exploring the practice of machine learning. We defined a model (a linear regression), and based about assumptions about how to get closer to ideal parameters, we used the technique of gradient descent to estimate the best possible

relationship between S and L . In order to provide a fair evaluation of the performance of this model, we kept a part of the dataset hidden from it while training. In Chapter 4, we will explore this last point in great depth, by introducing the concept of cross-validation, testing set, and performance evaluation.

References

- Brose, U., Ostling, A., Harrison, K. & Martinez, N.D. (2004). [Unified spatial scaling of species and their trophic interactions](#). *Nature*, 428, 167–171.
- Cohen, J.E. & Briand, F. (1984). Trophic links of community food webs. *Proc Natl Acad Sci U S A*, 81, 4105–4109.
- Innes, M. (2018). [Don't unroll adjoint: Differentiating SSA-form programs](#).
- MacDonald, A.A.M., Banville, F. & Poisot, T. (2020). [Revisiting the links-species scaling relationship in food webs](#). *Patterns*, 1.
- Martinez, N.D. (1992). [Constant connectance in community food webs](#). *The American Naturalist*, 139, 1208–1218.
- Morrison, M. & Morgan, M.S. (1999). [Models as mediating instruments](#). Cambridge University Press, pp. 10–37.

4. Cross-validation

In Chapter 2, we were very lucky. Because we applied an unsupervised method, we didn't really have a target to compare to the output. Whatever classification we got, we had to live with it. It was incredibly freeing. Sadly, in most applications, we will have to compare our predictions to data, and data are incredibly vexatious. In this chapter, we will develop intuitions on the notions of training, testing, and validation.

In a sense, we started thinking about these concepts in Chapter 3; specifically, we came up with a way to optimize the parameters of our model (*i.e.* of *training* our model) based on a series of empirical observations, and a criteria for what a “good fit” is. We further appraised the performance of our model by measuring the loss (our measure of how good the fit is) on a dataset that was not accessible during training, which we called the *testing* dataset. One issue with our approach in Chapter 3 was that we had to set aside one out of five observation for testing; in this chapter, we will explore more advanced techniques to perform cross-validation.

4.1. How can we split a dataset?

There is a much more important question to ask first: *why* do we split a dataset? In a sense, answering this question echoes the discussion we started in Section 3.4.4, because the purpose of splitting a dataset is to ensure we can train and evaluate it properly, in order to deliver the best possible model.

When a model is trained, it has learned from the data, we have tuned its hyper-parameters to ensure that it learned with the best possible conditions, and we have applied a measure of performance *after* the entire process is complete, to communicate how well we expect our model to work. These three tasks require three different datasets, and this is the purpose of splitting our data into groups.

4. Cross-validation

One of the issues when reading about splitting data is that the terminology can be muddy. For example, what constitutes a testing and validation set can largely be a matter of perspective. In many instances, testing and validation are used interchangeably, especially when there is a single model involved. Nevertheless, it helps to settle on a few guidelines here, before going into the details of what each dataset constitutes and how to assemble it.

The *training* instances are examples that are given to the model during the training process. This dataset has the least ambiguous definition. The training data is defined by subtraction, in a sense, as whatever is left of the original data after we set aside testing and validation sets.

The *testing* instances are used at the end of the process, to measure the performance of a trained model with tuned hyper-parameters. If the training data are the lectures, testing data are the final exam: we can measure the performance of the model on this dataset and report it as the model performance we can expect when applying the model to new data. There is a very important, chapter-long, caveat about this last point, related to the potential of information leak between datasets, which is covered in [?@sec-leakage](#).

The *validation* data are used in-between, as part of the training process. They are (possibly) a subset of the training data that we use internally to check the performance of the model, often in order to tune its hyper-parameters, or as a way to report on the over-fitting of the model during the training process.

The difference between testing and validation is largely a difference of *intent*. When we want to provide an *a posteriori* assessment of the model performance, the dataset we use to determine this performance is a testing dataset. When we want to optimize some aspect of the model, the data we use for this are the validation data. With this high-level perspective in mind, let's look at each of these datasets in turn. The differences between these three datasets are summarized in Table 4.1.

Table 4.1.: Overview of the three datasets used for training and cross-validation. Information in the “Data used for training” column refer to the data that have been used to train the model when calculating its performance.

Dataset	Trains	Purpose	Data used for training
Training	yes	train model	
Validation		validate during training	training data only

4.1. How can we split a dataset?

Dataset	Trains	Purpose	Data used for training
Testing		estimates of future performance	all except testing

4.1.1. Training

In data science (in applied machine learning in particular), we do not *fit* models. We *train* them. This is an important difference: training is an iterative process, that we can repeat, optimize, and tweak. The outcome of training and the outcome of fitting are essentially the same (a model that is parameterized to work as well as possible on a given dataset), but it is good practice to adopt the language of a field, and the language of data science emphasizes the different practices in model training.

Training, to provide a general definition, is the action of modifying the parameters of a model, based on knowledge of the data, and the error that results from using the current parameter values. In Chapter 3, for example, we saw how to train a linear model using the technique of gradient descent, based on a specific dataset, with a learning rate and loss function we picked based on trial and error. Our focus in this chapter is not on the methods we use for training, but on the data that are required to train a model.

Training a model is a process akin to rote learning: we will present the same input, and the same expected responses, many times over, and we will find ways for the error on each response to decrease (this is usually achieved by minimizing the loss function).

In order to initiate this process, we need an untrained model. Untrained, in this context, refers to a model that has not been trained *on the specific problem* we are addressing; the model may have been trained on a different problem (for example, we want to predict the distribution of a species based on a GLM trained on a phylogenetically related species). It is important to note that by “training the model”, what we really mean is “change the structure of the parameters until the output looks right”. For example, assuming a simple linear model like $c(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$, training this model would lead to changes in the values of β , but not to the consideration of a new model $c(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$. Comparing models is (often) the point of validation, which we will address later on.

4. Cross-validation

4.1.2. Validating

The easiest way to think about the validation dataset is by thinking about what it is *not* used for: training the model (this is the training set), and giving a final overview of the model expected performance (this is the testing set). The validation set is used for everything else (model selection, cross-validation, hyper-parameters tuning), albeit in a specific way. With the training set, we communicate the predictors and the labels to the model, and update the weights of the model in response. With the validation set, we communicate the predictors and the labels to the model, but we do *not* update the weights in response. All we care about during validation is the performance of the model on a problem it has not yet encountered during this specific round of training. If the training set is like attending a lecture, the validation set is formative feedback.

Of course, one issue with the creation of a validation set is that it needs to resemble the problem the model will have to solve in practice. We will discuss this more in depth in the following sections, but it is worth thinking about an example. Assume a model that classifies a picture as having either a black bear, or no black bear. Now, we can train this model using, for example, images from 10 camera traps that are situated in a forest. And we might want to validate with a camera trap that is in a zoo. In one of the enclosures. The one with a bear. A polar one.

The issue with this dataset as a validation dataset is that it does not matches the problem we try to solve in many different ways. First, we will have an excess of images with bears compared to our problem environment. Camera traps can have a large number of spurious activation, resulting in images without animals in them (Newey *et al.* 2015). Second, the data will come from very different environments (forest v. zoo). Finally, we are attempting to validate on something that is an entirely different species of bear. This sounds like an egregious case (it is), but it is easy to commit this type of mistake when our data get more complex than black bear, polar bear, no bear.

Validation is, in particular, very difficult when the dataset we use for training has extreme events (Bellocchi *et al.* 2010). Similarly, the efficiency of validation datasets can be limited if it reflects the same biases as the training data (Martinez-Meyer 2005). Recall that this validation dataset is used to decide on the ideal conditions to train the final model before testing (and eventually, deployment); it is, therefore, extremely important to get it right. A large number of techniques to split data (Goot 2021; Søgaard *et al.* 2021) use heuristics to minimize the risk of picking the wrong validation data.

4.2. The problem: cherry blossom phenology

4.1.3. Testing

The testing dataset is special. The model has *never* touched it. Not during training, and not for validation. For this reason, we can give it a very unique status: it is an analogue to data that are newly collected, and ready to be passed through the trained model in order to make a prediction.

The only difference between the testing set and actual new data is that, for the testing set, we know the labels. In other words, we can compare the model output to these labels, and this gives us an estimate of the model performance on future data. Assuming that this data selection was representative of the real data we will use for our model once it is trained, the performance on the validation set should be a good baseline for what to expect in production.

But this requires a trained model, and we sort of glossed over this step.

In order to come up with a trained model, it would be a strange idea not to use the validation data – they are, after all, holding information about the data we want to model! Once we have evaluated our model on the validation set, we can start the last round of training to produce the final model. We do this by training the model using everything *except* the testing data. This is an appropriate thing to do: because we have evaluated the model on the validation data, and assuming that it has a correct performance, we can expect that retraining the model on the validation data will not change the performance of the model.

4.2. The problem: cherry blossom phenology

The cherry blossom tree (*Prunus*) is renowned for its impressive bloom, which happens from March to April. The blooming, and associated festivals, are of particular cultural significance (Moriuchi & Basil 2019), and is therefore a cultural ecosystem service (Kosanic & Petzold 2020). Climate change has a demonstrable effect on the date of first bloom on *Prunus* species in Japan (Primack *et al.* 2009), which can affect the sustainability of cherry blossom festivals in the short term (Sakurai *et al.* 2011).

Long-term time series of the date of first bloom in Japan reveal that in the last decades, cherry blossom blooms earlier, which has been linked to, possibly, climate change and urbanization. *Prunus* species respond to environmental cues at the local level for their flowering (Mimet *et al.* 2009; Ohashi *et al.* 2011). The suspected causal

4. Cross-validation

mechanism is as follows: both global warming and urbanization lead to higher temperatures, which means a faster accumulation of degree days over the growing season, leading to an earlier bloom (Shi *et al.* 2017). Indeed, the raw data presented in Figure 4.1 show that trees bloom early when the temperatures are higher; the data for phenology have been collected by Aono & Kazui (2008), and the temperature reconstructions are from Aono & Saito (2009).

With these data in hand (day of year with the first bloom, and smoothed reconstructed temperature in March), we can start thinking about this hypothesis. But by contrast with our simple strategy in Chapter 3, this time, we will split our dataset into training, validation, and testing sets, as we discussed in the previous section. Yet there are many ways to split a dataset, and therefore before starting the analysis, we will have a look at a few of them.

4.3. Strategies to split data

Before seeing examples of strategies for cross-validation, it is important to consider the high-level perspective of the way we will perform the entire training sequence. First, we need to keep a testing dataset. Depending on the problem, it may be *feasible* or *desirable* to use an external testing dataset (Homeyer *et al.* 2022). In problems for which the volume of data is limited (the 99.99% of biodiversity applications that do not involve metagenomics or remote sensing), this is almost impossible, and therefore we need to resort to removing a proportion of the data. It means that collected data will never be used for training, which is not ideal, but what we gain in return is a fairer appraisal of the performance of the model, which is a really advantageous trade-off. When the testing data are removed, we can start splitting the rest of the data in training and validation sets. This can involve two broad categories of families: exhaustive splits (all data are used for training and evaluation), and non-exhaustive splits (the opposite; for once, the terminology makes sense!).

4.3.1. Holdout

The holdout method is what we used in Chapter 3, in which we randomly selected some observations to be part of the validation data (which was, in practice, a testing dataset in this example), and kept the rest to

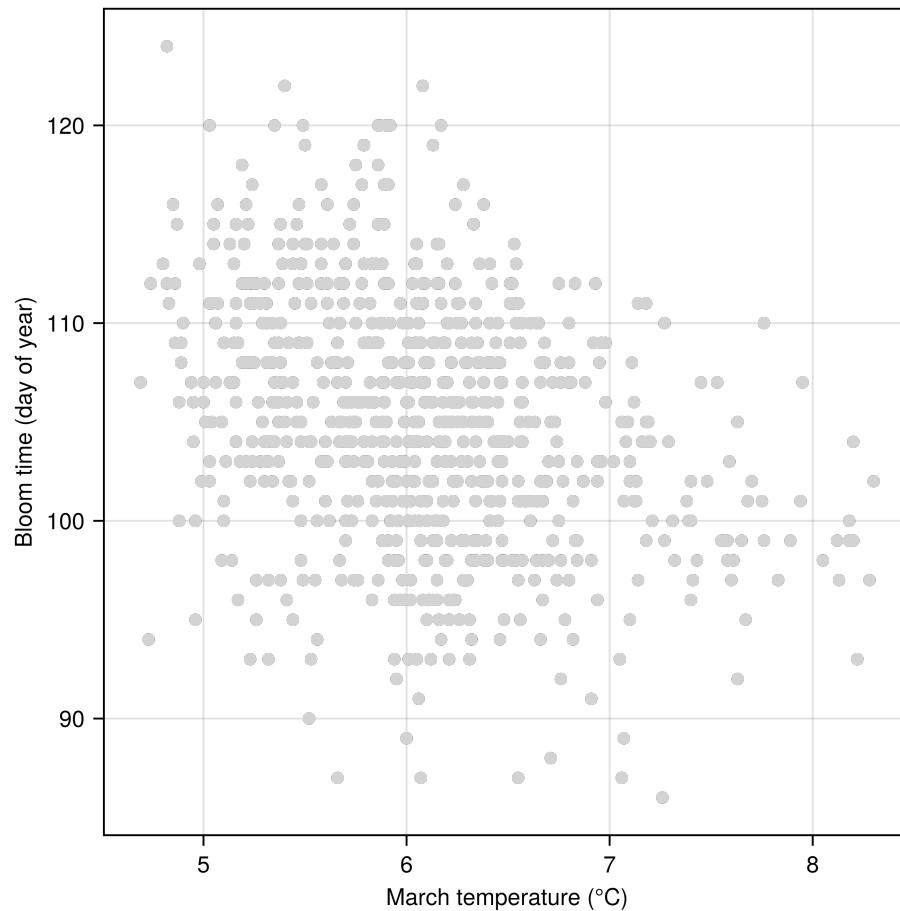


Figure 4.1.: The raw data show a negative relationship between the temperature in March, and the bloom time. This suggests that when the trees have accumulated enough temperature, they can bloom early. In a context of warming, we should therefore see earlier blooms with rising temperatures.

4. Cross-validation

serve as the training data. Holdout cross-validation is possibly the simplest technique, but it suffers from a few drawbacks.

The model is only trained for one split of the data, and similarly only evaluated for one split of the data. There is, therefore, a chance to sample a particularly bad combination of the data that lead to erroneous results. Attempts to quantify the importance of the predictors are likely to give particularly unstable results, as the noise introduced by picking a single random subset will not be smoothed out by multiple attempts.

In addition, as Hawkins *et al.* (2003) point out, holdout validation is particularly wasteful in data-limited settings, where there are fewer than hundreds of observations. The reason is that the holdout dataset will *never* contribute to training, and assuming the data are split 80/20, one out of five observations will not contribute to the model. Other cross-validation schemes presented in this section will allow observations to be used both for training and validation.

4.3.2. Leave-p-out

In leave- p -out cross-validation (LpOCV), starting from a dataset on n observations, we pick p at random to serve as validation data, and $n - p$ to serve as the training dataset. This process is then repeated *exhaustively*, which is to say we split the dataset in every possible way that gives p and $n - p$ observations, for a set value of p . The model is then trained on the $n - p$ observations, and validated on the p observations for validation, and the performance (or loss) is averaged to give the model performance before testing.

Celisse (2014) points out that p has to be large enough (relative to the sample size n) to overcome the propensity of the model to overfit on a small training dataset. One issue with LpOCV is that the number of combinations is potentially very large. It is, in fact, given by the binomial coefficient $\binom{n}{p}$, which gets unreasonably large even for small datasets. For example, running LpOCV on $n = 150$ observations, leaving out $p = 10$ for validation every time, would require to train the model about 10^{15} times. Assuming we can train the model in 10^{-3} seconds, the entire process would require 370 centuries.

Oh well.

4.3.3. Leave-one-out

The leave-one-out cross-validation (LOOCV) is a special case of LpOCV with $p = 1$. Note that it is a lot faster to run than LpOCV, because $\binom{n}{1} = n$, and so the validation step runs in $\mathcal{O}(n)$ (LpOCV runs in $\mathcal{O}(n!)$). LOOCV is also an *exhaustive* cross-validation technique, as every possible way to split the dataset will be used for training and evaluation.

4.3.4. k-fold

One of the most frequent cross-validation scheme is k-fold cross-validation. Under this approach, the dataset is split into k equal parts (and so when $k = n$, this is also equivalent to LOOCV). Like with LOOCV, one desirable property of k-fold cross-validation is that each observation is used *exactly* one time to evaluate the model, and *exactly* $k - 1$ times to train it.

But by contrast with the holdout validation approach, *all* observations are used to train the model.

When the data have some specific structure, it can be a good thing to manipulate the splits in order to maintain this structure. For example, Bergmeir & Benítez (2012) use temporal blocks for validation of time series, and retain the last part of the series for testing (we illustrate this in Figure 4.2). For spatial data, Hijmans (2012) suggests the use of a null model based on distance to training sites to decide on how to split the data; Valavi *et al.* (2018) have designed specific k-fold cross-validation schemes for species distribution models. These approaches all belong to the family of *stratified* k-fold cross-validation (Zeng & Martinez 2000).

The appropriate value of k is often an unknown. It is common to use $k = 10$ as a starting point (tenfold cross-validation), but other values are justifiable based on data volume, or complexity of the model training, to name a few.

4.3.5. Monte-Carlo

One limitation of k-fold cross-validation is that the number of splits is limited by the amount of observations, especially if we want to ensure that there are enough samples in the validation data. To compensate for this,

4. Cross-validation

Figure 4.2.: An illustration of a series of folds on a timeseries. The grey data are used for training, the green data for validation, and the purple data are kept for testing. Note that in this context, we sometimes use the future to validate on the past (look at the first fold!), but this is acceptable for reasons explained in the text.



4.4. Application: when do cherry blossom bloom?

Monte-Carlo cross-validation is essentially the application (and averaging) of holdout validation an arbitrary number of times. Furthermore, the training and validation datasets can be constructed in order to account for specific constraints in the dataset, giving more flexibility than k-fold cross-validation (Roberts *et al.* 2017). When the (computational) cost of training the model is high, and the dataset has specific structural constraints, Monte-Carlo cross-validation is a good way to generate data for hyperparameters tuning.

One issue with Monte-Carlo cross-validation is that we lose the guarantee that every observation will be used for training at least once (and similarly for validation). Trivially, this becomes less of an issue when we increase the number of replications, but then this suffers from the same issues as LpOCV, namely the unreasonable computational requirements.

4.4. Application: when do cherry blossom bloom?

The model we will train for this section is really simple: $\text{bloom day} = m \times \text{temperature} + b$. This is a linear model, and one with a nice, direct biological interpretation: the average (baseline) day of bloom is b , and each degree of temperature expected in March adds m days to the bloom date. At this point, we *might* start thinking about the distribution of the response, and what type of GLM we should used, but no. Not today. Today, we want to iterate quickly, and so we will start with a model that is exactly as simple as it needs to be: this is, in our case, linear regression.

At this point, we may be tempted to think a little more deeply about the variables and the structure of the model, to express the bloom day as a departure from the expected value, and similarly with the temperature, using for example the z-score. This is a transformation we will apply starting from Chapter 5, but in order to apply it properly, we need to consider some elements that will be introduced in [?@sec-leakage](#). For this reason, we will not apply any transformation to the data yet; feel free to revisit this exercise after reading through [?@sec-leakage](#).

This approach (start from a model that is suspiciously simple) is a good thing, for more than a few reasons. First, it gives us a baseline to compare more complicated models against. Second, it means that we do not need to focus on the complexity of the code (and the model) when building a pipeline for the analysis. Finally, and most importantly, it gives us a result very rapidly, which enables a loop of iterative model refinement on a very

4. *Cross-validation*

short timescale. Additionally, at least for this example, the simple models often work well enough to support a discussion of the model and training process.

4.4.1. Performance evaluation

We can visualize the results of our model training and assessment process. These results are presented in Figure 4.3 (as well as in Table 4.2, if you want to see the standard deviation across all splits), and follow the same color-coding convention we have used so far. All three loss measures presented here express their loss in the units of the response variable, which in this case is the day of the year where the bloom was recorded. These results show that our trained model achieves a loss of the order of a day or two in the testing data, which sounds really good!

Yet it is important to contextualize these results. What does it means for our prediction to be correct plus or minus two days? There are at least two important points to consider.

Table 4.2.: Summary of the model performance after cross-validation.

Dataset	Measure	Loss (avg.)	Loss (std. dev.)
Testing	MAE	1.696	
Training	MAE	2.2397	0.0482364
Validation	MAE	2.26331	0.421513
Testing	MBE	0.0971036	
Training	MBE	9.8278e-15	1.15597e-14
Validation	MBE	0.000419595	0.910229
Testing	MSE	4.49123	
Training	MSE	8.04855	0.32487
Validation	MSE	8.24897	2.93094
Testing	RMSE	2.11925	
Training	RMSE	2.83648	0.0570941
Validation	RMSE	2.82514	0.545232

4.4. Application: when do cherry blossom bloom?

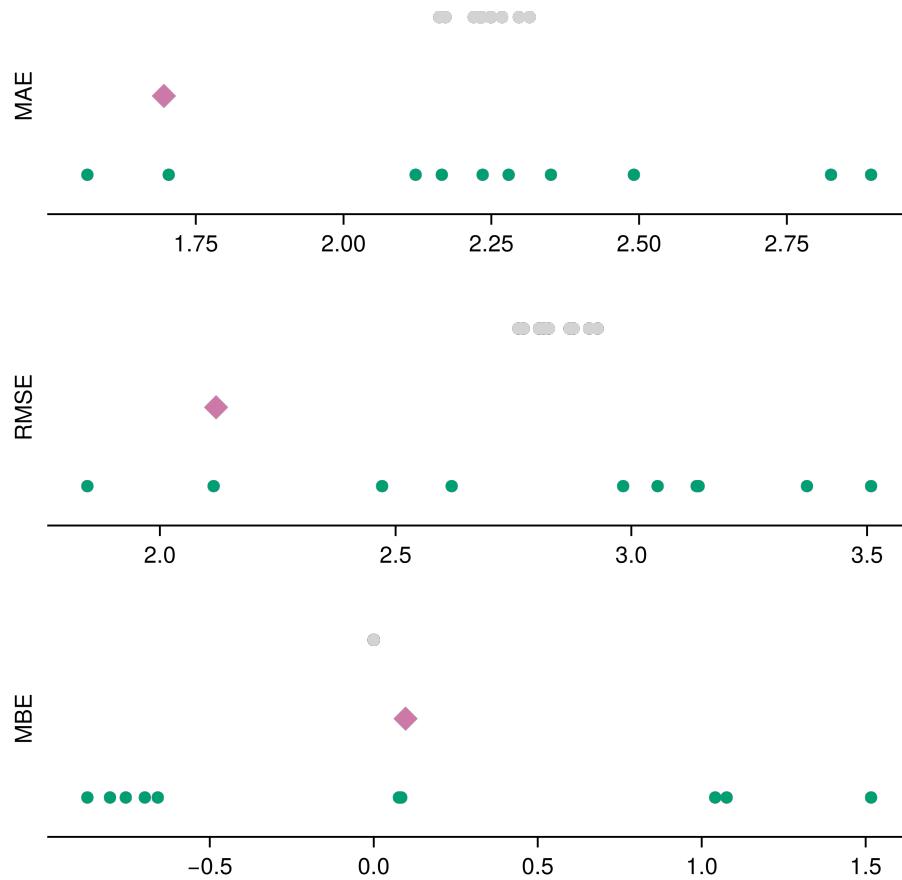


Figure 4.3.: Visualisation of the model performance for three loss functions (MAE, RMSE, MBE, as defined in Table 3.1). The colors are the same as in Figure 4.2, i.e. grey for the training data, green for the validation data, and purple for the testing data.

4. Cross-validation

First, what are we predicting? Our response variable is not *really* the day of the bloom, but is rather a smoothed average looking back some years, and looking ahead some years too. For this reason, we are removing a lot of the variability in the underlying time series. This is not necessarily a bad thing, especially if we are looking for a trend at a large temporal scale, but it means that we should not interpret our results at a scale lower than the duration of the window we use for averaging.

Second, what difference *does* a day make? Figure 4.1 shows that most of the days of bloom happen between day-of-year 100 and day-of-year 110. Recall that the MAE is measured by taking the average absolute error – a mistake of 24 hours is 10% of this interval! This is an example of how thinking about the units of the loss function we use for model evaluation can help us contextualize the predictions, and in particular how actionable they can be.

4.4.2. Model predictions

The predictions of our model are presented in Figure 4.4; these are the predictions of the *final* model, that is, the model that we trained on everything *except* the testing data, and for which we can get the performance by looking at Figure 4.3.

The question we now need to answer is: is our model doing a good job? We can start thinking about this question in a very qualitative way: yes, it does a good job at drawing a line that, through time, goes right through the original data more often than it doesn't. As far as validation goes, it maybe underestimates the drop in the response variable (it predicts the bloom a little later), but maybe there are long-term effects, expressed over the lifetime of the tree (the first bloom usually takes place after 6 or 7 growth seasons), that we do not account for.

Think about the structure of linear models. Can we use information about the previous years in our model? Would there be a risk associated to adding more parameters?

Our model tends to smooth out some of the variation; it does not predict bloom dates before day of year 100, or after day of year 108, although they do happen. This may not be a trivial under-prediction: some of these cycles leading to very early/late bloom can take place over a century, meaning that our model could be consistently wrong (which is to say, wrong with the same bias) for dozens of years in a row.

4.4. Application: when do cherry blossom bloom?

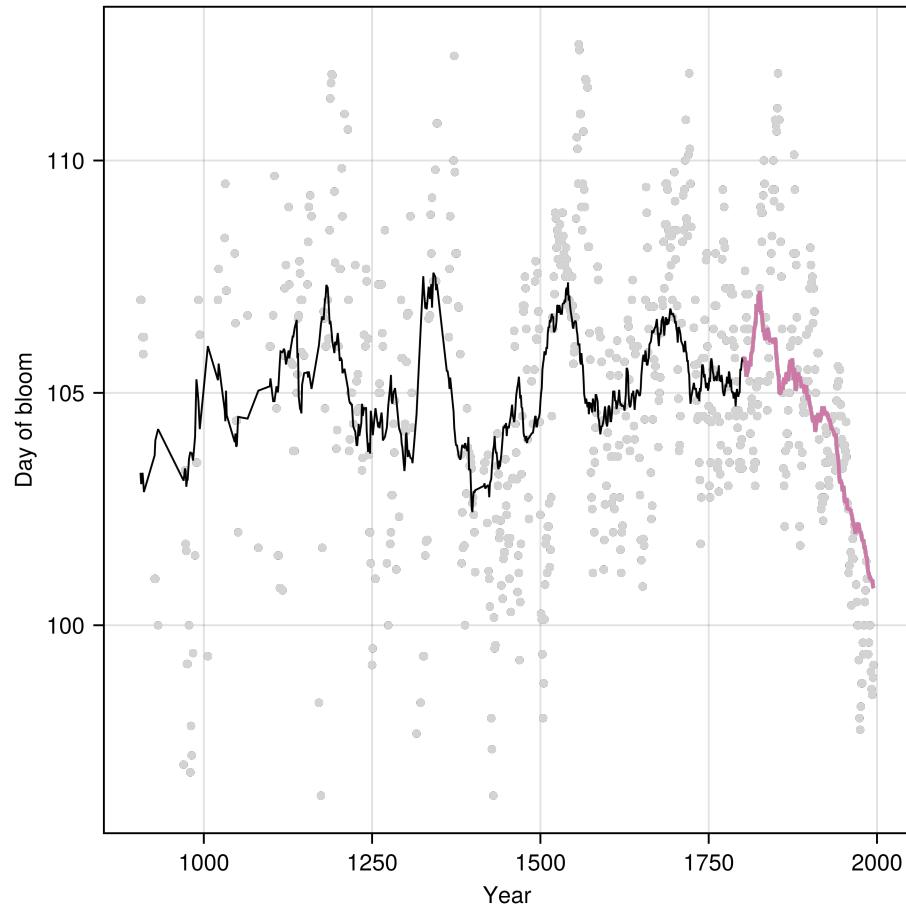


Figure 4.4.: Overview of the fit of the final model (trained on all the training examples), visualized as the time series. Note that the year was not used as a variable in the model. The purple part of the prediction corresponds to the prediction of the model for the testing data, which are zoomed-in on in Figure 4.5. Although the model captures the cycles reasonably well, it tends to smooth out a lot of extreme events.

4. Cross-validation

4.4.3. Is our model good, then?

The answer is, it depends. Models are neither good, nor bad. They are either fit, or unfit, for a specific purpose.

If the purpose is to decide when to schedule a one-day trip to see the cherry blossom bloom, our model is not really fit – looking at the predictions, it gets within a day of the date of bloom (but oh, by the way, this is an average over almost a decade!) about 15% of the time, which jumps up to almost 30% if you accept a two-days window of error.

If the purpose is to look at long-time trends in the date of bloom, then our model actually works rather well. It does under-estimate the amplitude of the cycles, but not by a large amount. In fact, we could probably stretch the predictions a little, applying a little correction factor, and have a far more interesting model.

We will often be confronted to this question when working with prediction. There is not really a criteria for “good”, only a series of compromises and judgment calls about “good enough”. This is important. It reinforces the imperative of keeping the practice of data science connected to the domain knowledge, as ultimately, a domain expert will have to settle on whether to use a model or not.

4.5. Conclusion

In this chapter, we trained a linear regression model to predict the day of bloom of cherry blossom trees based on the predicted temperature in March. Although the model makes a reasonable error (of the order of a few days), a deeper investigation of the amplitude of this error compared to the amplitude of the response variable, and of the comparison of extreme values in the prediction and in the data, led us to a more cautious view about the usefulness of this model. In practice, if we really wanted to solve this problem, this is the point where we would either add variables, or try another regression algorithm, or both.

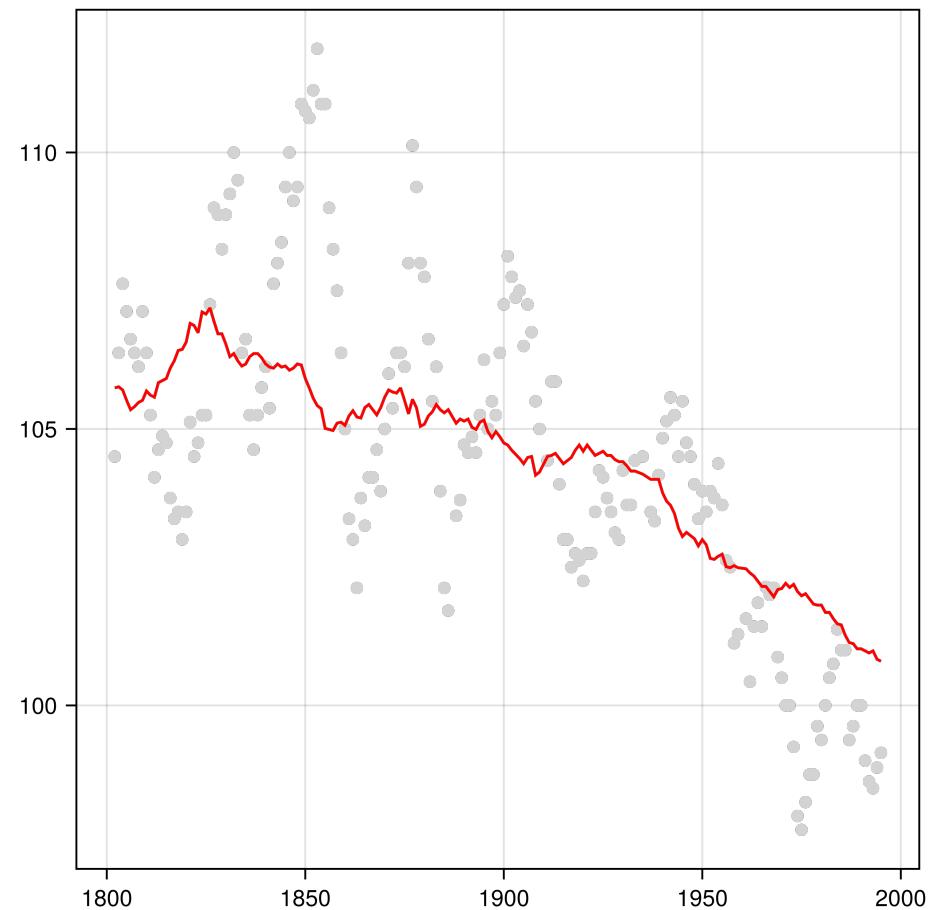


Figure 4.5.: Overview of the model predictions on the testing data. Note that the model still smoothes out some of the extreme values. More importantly, it seems that it is under-estimating the sharp decline in the day of first bloom that happens starting in 1950; this suggests that the model is not adequately capturing important processes shaping the data.

4. Cross-validation

References

- Aono, Y. & Kazui, K. (2008). Phenological data series of cherry tree flowering in Kyoto, Japan, and its application to reconstruction of springtime temperatures since the 9th century. *International Journal of Climatology*, 28, 905–914.
- Aono, Y. & Saito, S. (2009). Clarifying springtime temperature reconstructions of the medieval period by gap-filling the cherry blossom phenological data series at Kyoto, Japan. *International Journal of Biometeorology*, 54, 211–219.
- Bellocchi, G., Rivington, M., Donatelli, M. & Matthews, K. (2010). Validation of biophysical models: issues and methodologies. A review. *Agronomy for Sustainable Development*, 30, 109–130.
- Bergmeir, C. & Benítez, J.M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213.
- Celisse, A. (2014). Optimal cross-validation in density estimation with the $\$l^2\$$ -loss. *The Annals of Statistics*, 42.
- Goot, R. van der. (2021). We need to talk about train-dev-test splits. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.
- Hawkins, D.M., Basak, S.C. & Mills, D. (2003). Assessing Model Fit by Cross-Validation. *Journal of Chemical Information and Computer Sciences*, 43, 579–586.
- Hijmans, R.J. (2012). Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. *Ecology*, 93, 679–688.
- Homeyer, A., Geißler, C., Schwen, L.O., Zakrzewski, F., Evans, T., Strohmenger, K., et al. (2022). Recommendations on compiling test datasets for evaluating artificial intelligence solutions in pathology. *Modern Pathology*, 35, 1759–1769.
- Kosanic, A. & Petzold, J. (2020). A systematic review of cultural ecosystem services and human wellbeing. *Ecosystem Services*, 45, 101168.
- Martinez-Meyer, E. (2005). Climate change and biodiversity: Some considerations in forecasting shifts in species' potential distributions. *Biodiversity Informatics*, 2.
- Mimet, A., Pellissier, V., Quénol, H., Aguejdad, R., Dubreuil, V. & Rozé, F. (2009). Urbanisation induces early flowering: evidence from *Platanus acerifolia* and *Prunus cerasus*. *International Journal of Biometeorology*, 53, 287–298.
- Moriuchi, E. & Basil, M. (2019). The Sustainability of Ohanami Cherry Blossom Festivals as a Cultural Icon.

- Sustainability*, 11, 1820.
- Newey, S., Davidson, P., Nazir, S., Fairhurst, G., Verdicchio, F., Irvine, R.J., *et al.* (2015). Limitations of recreational camera traps for wildlife management and conservation research: A practitioner's perspective. *Ambio*, 44, 624–635.
- Ohashi, Y., Kawakami, H., Shigeta, Y., Ikeda, H. & Yamamoto, N. (2011). The phenology of cherry blossom (*Prunus yedoensis* "Somei-yoshino") and the geographic features contributing to its flowering. *International Journal of Biometeorology*, 56, 903–914.
- Primack, R.B., Higuchi, H. & Miller-Rushing, A.J. (2009). The impact of climate change on cherry trees and other species in Japan. *Biological Conservation*, 142, 1943–1949.
- Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., *et al.* (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, 40, 913–929.
- Sakurai, R., Jacobson, S.K., Kobori, H., Primack, R., Oka, K., Komatsu, N., *et al.* (2011). Culture and climate change: Japanese cherry blossom festivals and stakeholders' knowledge and attitudes about global climate change. *Biological Conservation*, 144, 654–658.
- Shi, P., Chen, Z., Reddy, G.V.P., Hui, C., Huang, J. & Xiao, M. (2017). Timing of cherry tree blooming: Contrasting effects of rising winter low temperatures and early spring temperatures. *Agricultural and Forest Meteorology*, 240-241, 78–89.
- Søgaard, A., Ebert, S., Bastings, J. & Filippova, K. (2021). We need to talk about random splits. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.
- Valavi, R., Elith, J., Lahoz-Monfort, J.J. & Guillera-Arroita, G. (2018). block CV : An r package for generating spatially or environmentally separated folds for k -fold cross-validation of species distribution models. *Methods in Ecology and Evolution*, 10, 225–232.
- Zeng, X. & Martinez, T.R. (2000). Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, 12, 1–12.

5. 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 [?@sec-predictors](#), we will explore techniques to fine-tune the set of variables we use for prediction; in [?@sec-tuning](#), 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.

5.1. 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 Section 4.4.3), has been covered in the SDM literature (Guillera-Arroita *et al.* 2015). In these

5. Supervised classification

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 $(\text{lat.}, \text{lon.}, y)$, 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 $x^T = [\text{temp. } \text{precip.}]$. This specific task is called classification, and we will now introduce some elements of theory.

5.2. What is classification?

Classification is the prediction of a qualitative response. In Chapter 2, 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 or 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.

5.2.1. 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 `?@sec-predictors-curse`, simply adding more predictors is not the right thing to do.

In Figure 5.1, 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 `?@sec-predictors`), 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

5. Supervised classification

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.

5.2.2. 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

$$\begin{pmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{pmatrix}. \quad (5.1)$$

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

$$\begin{pmatrix} |\hat{+}, +| & |\hat{+}, -| \\ |\hat{-}, +| & |\hat{-}, -| \end{pmatrix}. \quad (5.2)$$

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 5.2.4 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 is 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.

5.2. What is classification?

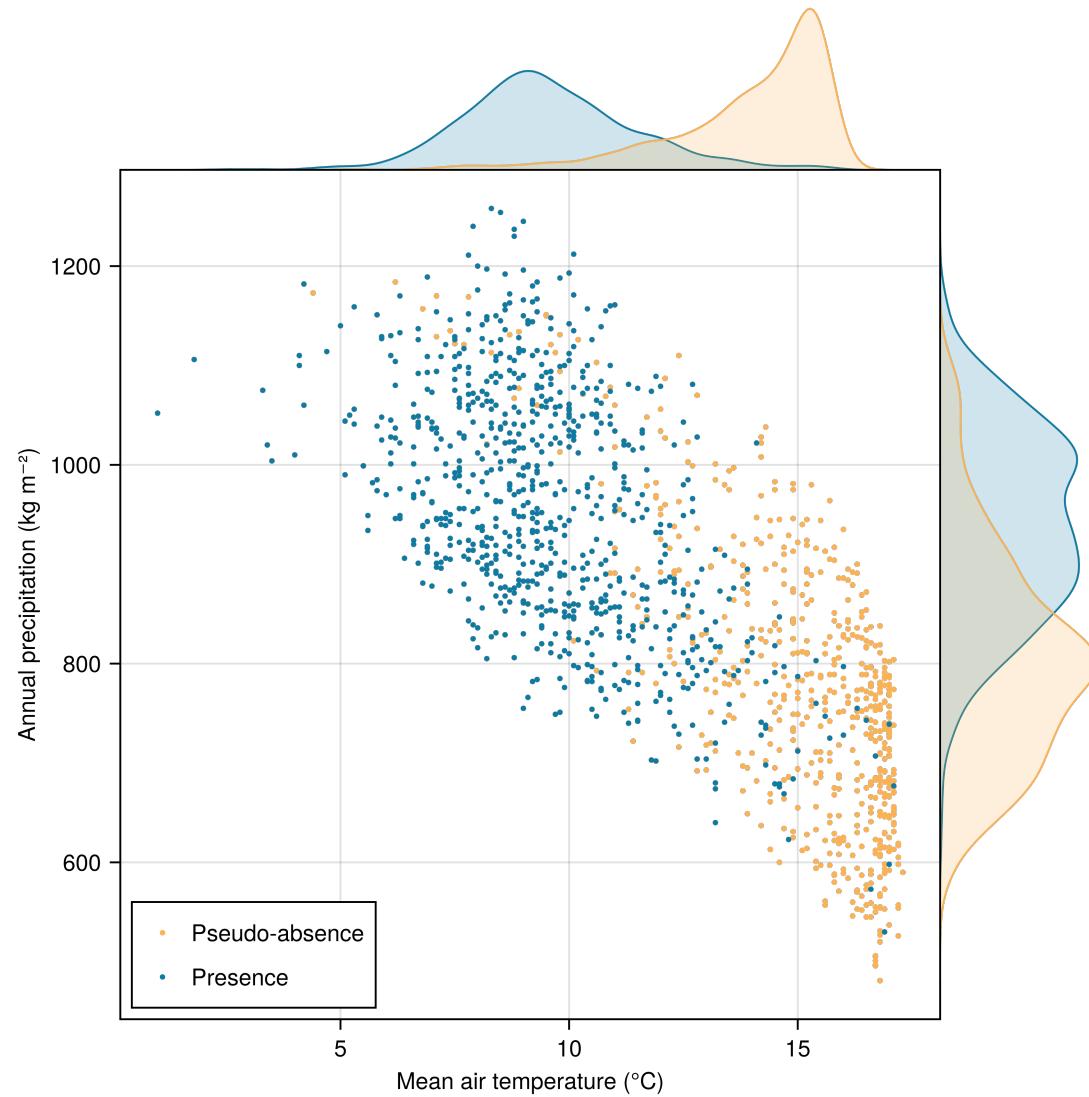


Figure 5.1.: This figures show the separability of the presences (orange) and pseudo-absences (grey) on the temperature and precipitation dimensions.

5. Supervised classification

5.2.3. 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.

5.2.3.1. 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

$$\begin{pmatrix} b^2 & b(1-b) \\ (1-b)b & (1-b)^2 \end{pmatrix}. \quad (5.3)$$

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 5.2.4 : 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.

5.2.3.2. 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:

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

5.2.3.3. 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 5.2.4).

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.

5. Supervised classification

5.2.4. 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.

5.3. 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).

5.3.1. How the NBC works

In Figure 5.1, 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 $P(8^\circ\text{C}|+) > P(8^\circ\text{C}|-)$. It would appear that we are looking at the problem in the wrong way, because we are really interested in $P(+|8^\circ\text{C})$, 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

$$P(+|x) = \frac{P(+)}{P(x)} P(x|+), \quad (5.4)$$

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 5.4, 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.

5. Supervised classification

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:

$$\hat{y} = \operatorname{argmax}_j P(\mathbf{c}_j) \prod_i P(\mathbf{x}_i|\mathbf{c}_j). \quad (5.5)$$

In other words, whichever class gives the higher score, is what the NBC will recommend for this instance \mathbf{x} . In **?@sec-tuning**, 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.

5.3.2. 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 entirely 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).

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 Chapter 3, 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.

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.

5.4. Application: a baseline model of the Corsican nuthatch

In this section, we will have a look at the temperature and precipitation data from Figure 5.1, 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.

5.4.1. Training and validation strategy

To evaluate our model, as we discussed in Chapter 4, 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 `?@sec-tuning`). 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 Section 4.3.5), 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).

5.4.2. 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

5.4. Application: a baseline model of the Corsican nuthatch

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 `?@sec-tuning`, and eventually explain why MCC is the most appropriate for classification evaluation).

The MCC is defined as

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

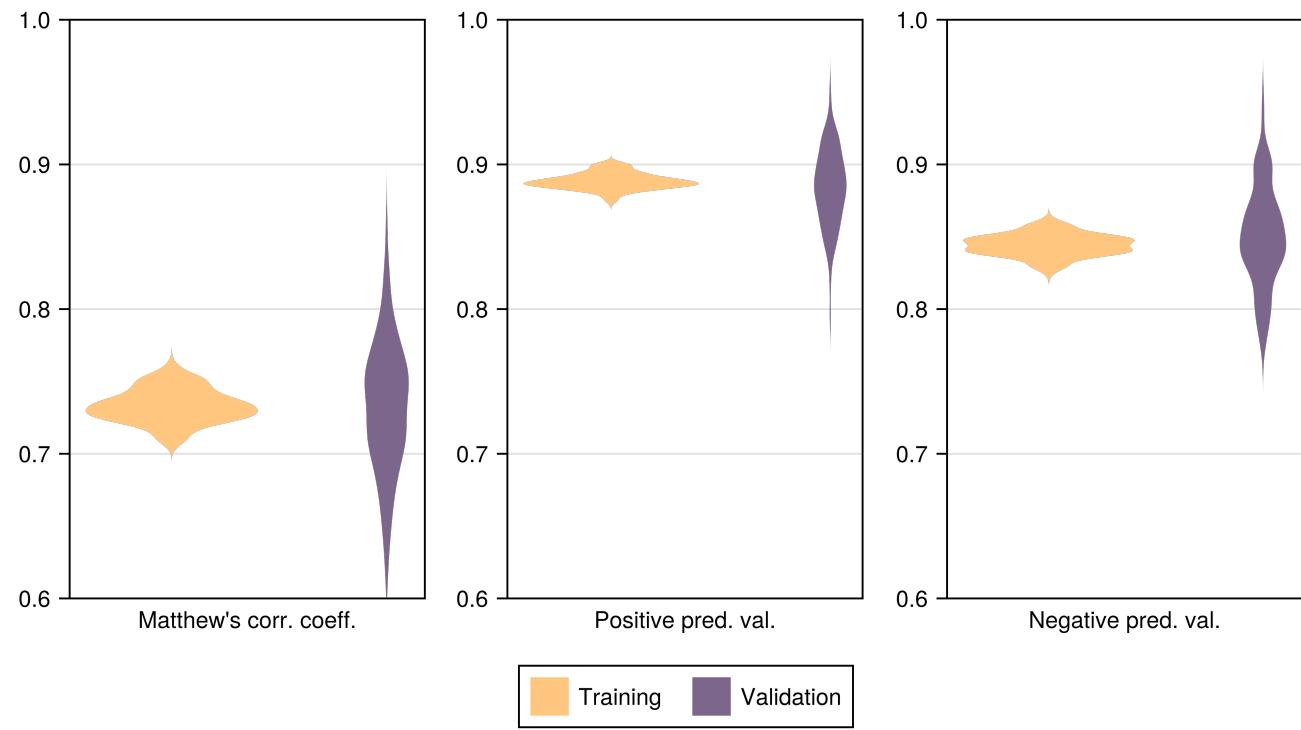
The MCC is a correlation coefficient. Specifically, it is the Pearson product-moment correlation on a contingency table, where the contingency table is the confusion table (Powers 2020). Therefore, it returns values in $[-1, 1]$, which can be interpreted as every other correlation value. A negative value 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 false 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.

5. Supervised classification

Figure 5.2.: Overview of the scores for the Matthew's correlation coefficient, as well as the positive and negative predictive values.



5.4. Application: a baseline model of the Corsican nuthatch

The output of cross-validation is given in Figure 5.2 (and compared to the no-skill classifier in Table 5.1). 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 5.1.: Overview of the data presented in Figure 5.2, compared to the null classifiers from Section 5.2.3. 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	0.57	0.57	0.43
NPV	0.84	0.85	0.43	0.57		0.43
PPV	0.89	0.88	0.57	0.57	0.57	
MCC	0.73	0.73	0.00	0.15	0.00	0.00

5.4.3. 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 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.

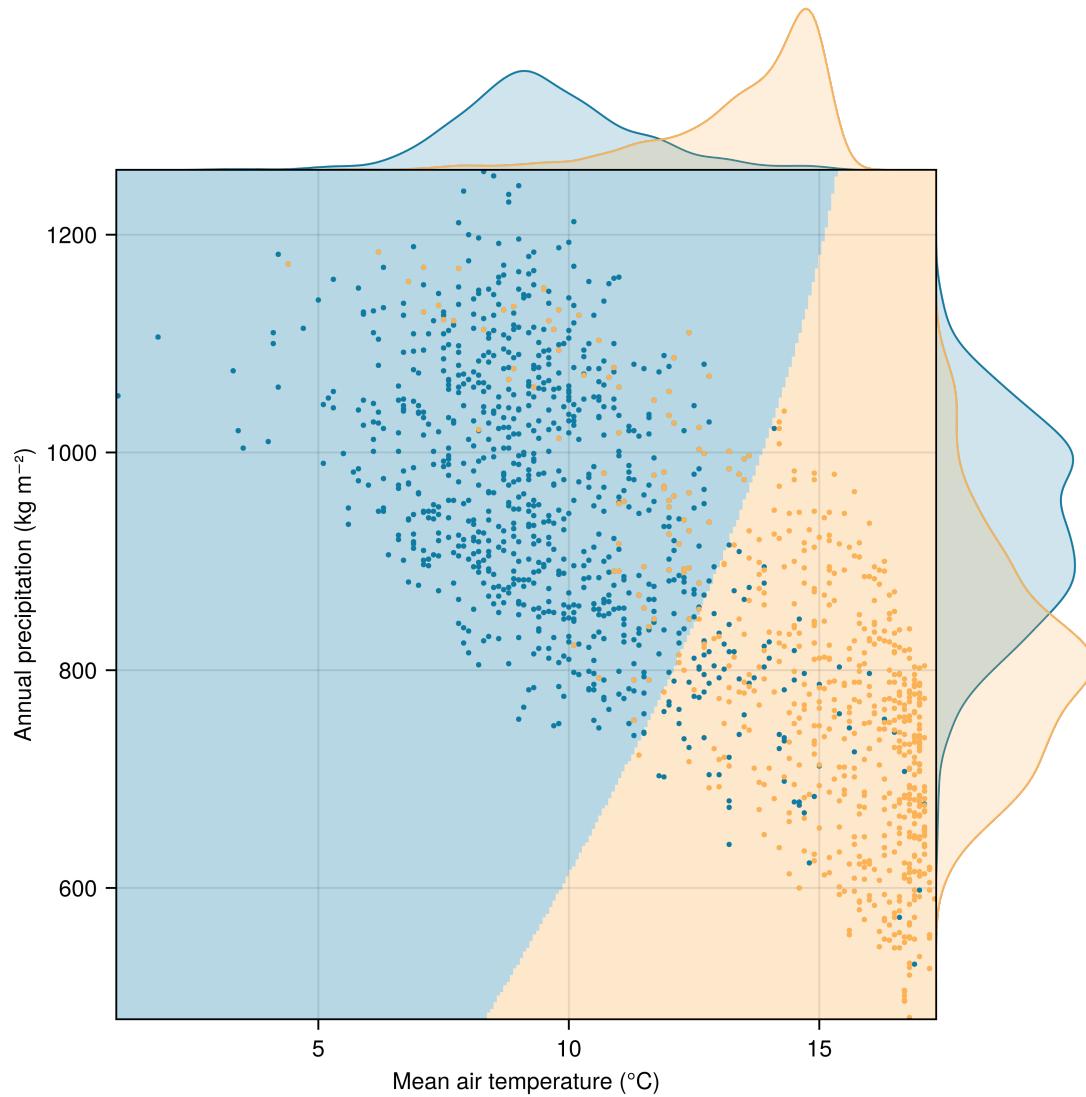
The output of this simulation is given in Figure 5.3. Of course, in a model with more features, we would need to adapt our visualisations, but because we only use two features here, this image actually gives us a complete understanding of the model decision process. Think of it this way: even if we lose 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?

5. Supervised classification

Figure 5.3.: 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 independence between the features means that we would not see, for example, an ellipse.



5.4. Application: a baseline model of the Corsican nuthatch

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 5.3 with, for example, the distribution of the raw data presented in Figure 5.1. 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.

5.4.4. 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 5.4. 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.

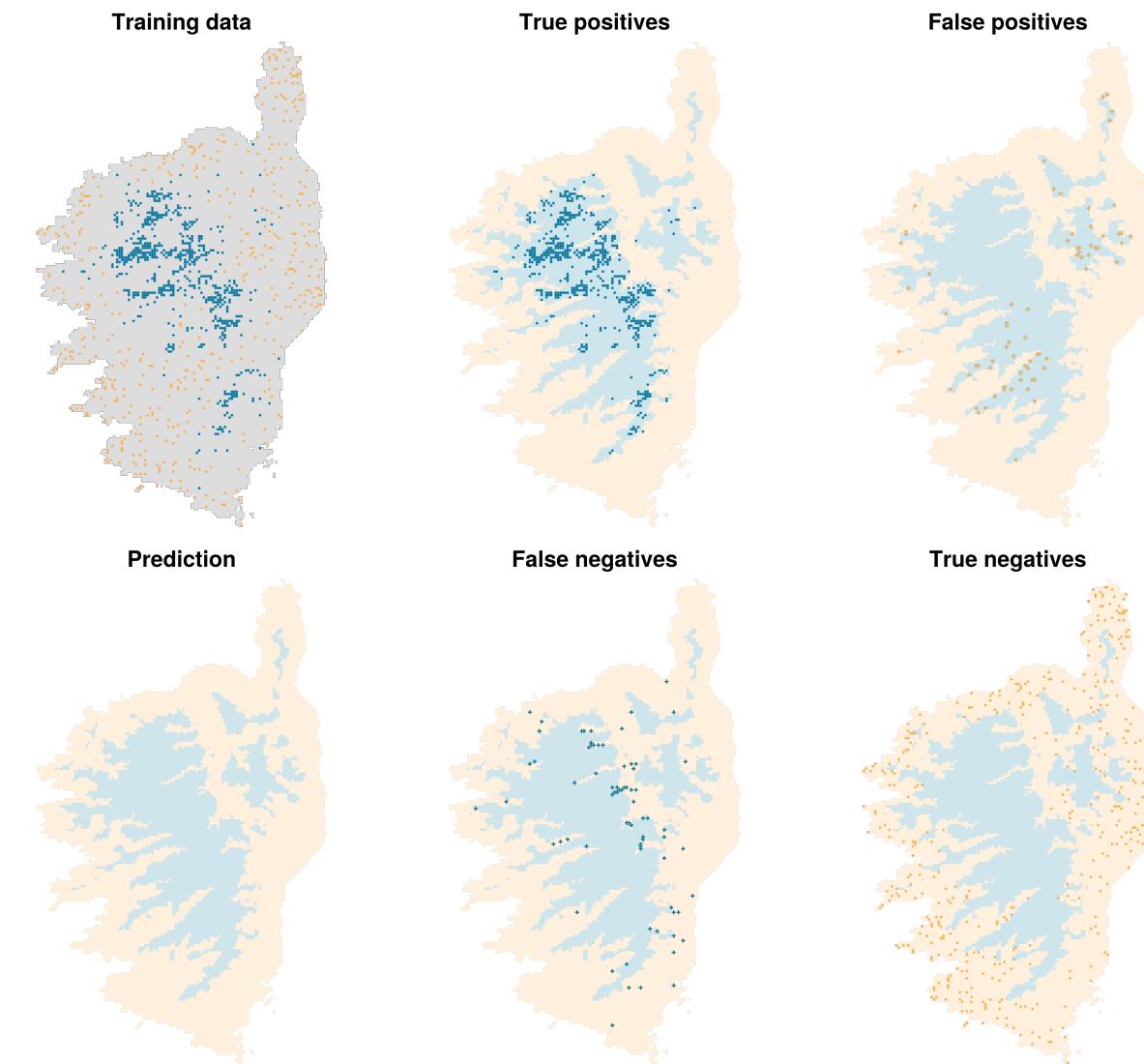
5.4.5. What is an acceptable model?

When comparing the prediction to the spatial distribution of occurrences (Figure 5.4), 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

5. Supervised classification

Figure 5.4.: Occurrence 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.

5.5. 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 5.2), looked at the decisions that were being made by the trained model (Figure 5.3), and finally mapped the prediction and their associated errors in space (Figure 5.4). 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. & Ghazzi, 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.

5. Supervised classification

- 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.

A. Instructor notes

References

- Abbood, A., Ullrich, A., Busche, R. & Ghazzi, S. (2020). EventEpi—A natural language processing framework for event-based surveillance. *PLOS Computational Biology*, 16, e1008277.
- Aloise, D., Deshpande, A., Hansen, P. & Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75, 245–248.
- 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.
- Aono, Y. & Kazui, K. (2008). Phenological data series of cherry tree flowering in Kyoto, Japan, and its application to reconstruction of springtime temperatures since the 9th century. *International Journal of Climatology*, 28, 905–914.
- Aono, Y. & Saito, S. (2009). Clarifying springtime temperature reconstructions of the medieval period by gap-filling the cherry blossom phenological data series at Kyoto, Japan. *International Journal of Biometeorology*, 54, 211–219.
- 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)*.
- Bellocchi, G., Rivington, M., Donatelli, M. & Matthews, K. (2010). Validation of biophysical models: issues and methodologies. A review. *Agronomy for Sustainable Development*, 30, 109–130.
- Bergmeir, C. & Benítez, J.M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213.
- Berteaux, D. (2014). *Changements climatiques et biodiversité du Québec*. Presses de l'Université du Québec.

A. Instructor notes

- 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.
- Brose, U., Ostling, A., Harrison, K. & Martinez, N.D. (2004). [Unified spatial scaling of species and their trophic interactions](#). *Nature*, 428, 167–171.
- Celisse, A. (2014). [Optimal cross-validation in density estimation with the \$\\$l^2\\$\$ -loss](#). *The Annals of Statistics*, 42.
- 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., Raunkjaer, C., Gilbert-Carter, H., Tansley, A.G. & Fausboll. (1935). [The life forms of plants and statistical plant geography](#). *The Journal of Ecology*, 23, 247.
- Cohen, J.E. & Briand, F. (1984). [Trophic links of community food webs](#). *Proc Natl Acad Sci U S A*, 81, 4105–4109.
- Cooney, C.R., He, Y., Varley, Z.K., Nouri, L.O., Moody, C.J.A., Jardine, M.D., et al. (2022). [Latitudinal gradients in avian colourfulness](#). *Nature Ecology & Evolution*, 6, 622–629.
- Cooper, N., Bond, A.L., Davis, J.L., Portela Miguez, R., Tomsett, L. & Helgen, K.M. (2019). [Sex biases in bird and mammal natural history collections](#). *Proceedings of the Royal Society B: Biological Sciences*, 286, 20192025.
- Dansereau, G., Barros, C. & Poisot, T. (2024). [Spatially explicit predictions of food web structure from regional-level data](#). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 379.
- Davies, D.L. & Bouldin, D.W. (1979). [A cluster separation measure](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1, 224–227.
- 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](#).
- Dove, S., Böhm, M., Freeman, R., Jellesmark, S. & Murrell, D.J. (2023). [A user-friendly guide to using distance](#)

- measures to compare time series in ecology. *Ecology and Evolution*, 13.
- Dunn, J.C. (1974). Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 4, 95–104.
- 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.
- 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.
- Gonzalez, A., Vihervaara, P., Balvanera, P., Bates, A.E., Bayraktarov, E., Bellingham, P.J., et al. (2023). A global biodiversity observing system to unite monitoring and guide action. *Nature Ecology & Evolution*.
- Goot, R. van der. (2021). We need to talk about train-dev-test splits. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.
- 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.
- Gray, R. (1984). Vector quantization. *IEEE ASSP Magazine*, 1, 4–29.
- 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.
- Hawkins, D.M., Basak, S.C. & Mills, D. (2003). Assessing Model Fit by Cross-Validation. *Journal of Chemical Information and Computer Sciences*, 43, 579–586.
- Hijmans, R.J. (2012). Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. *Ecology*, 93, 679–688.
- Homeyer, A., Geißler, C., Schwen, L.O., Zakrzewski, F., Evans, T., Strohmenger, K., et al. (2022). Recommendations on compiling test datasets for evaluating artificial intelligence solutions in pathology. *Modern Pathology*, 35, 1759–1769.
- Horst, A.M., Hill, A.P. & Gorman, K.B. (2020). *Allisonhorst/palmerpenguins: v0.1.0*. Zenodo.

A. Instructor notes

- Innes, M. (2018). Don't unroll adjoint: Differentiating SSA-form programs.
- 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.
- Kennedy, S. & Burbach, M. (2020). Great Plains Ranchers Managing for Vegetation Heterogeneity: A Multiple Case Study. *Great Plains Research*, 30, 137–148.
- Kosanic, A. & Petzold, J. (2020). A systematic review of cultural ecosystem services and human wellbeing. *Ecosystem Services*, 45, 101168.
- 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. & Gauthier, O. (2014). Statistical methods for temporal and space-time analysis of community composition data. *Proceedings of the Royal Society B: Biological Sciences*, 281, 20132728.
- Luccioni, A.S. & Rolnick, D. (2023). Bugs in the data: How ImageNet misrepresents biodiversity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 14382–14390.
- MacDonald, A.A.M., Banville, F. & Poisot, T. (2020). Revisiting the links-species scaling relationship in food webs. *Patterns*, 1.
- Martinez, N.D. (1992). Constant connectance in community food webs. *The American Naturalist*, 139, 1208–1218.
- Martinez-Meyer, E. (2005). Climate change and biodiversity: Some considerations in forecasting shifts in species' potential distributions. *Biodiversity Informatics*, 2.
- McFeeters, S. (2013). Using the Normalized Difference Water Index (NDWI) within a Geographic Information System to Detect Swimming Pools for Mosquito Abatement: A Practical Approach. *Remote Sensing*, 5, 3544–3561.
- Mimet, A., Pellissier, V., Quénol, H., Aguejdad, R., Dubreuil, V. & Rozé, F. (2009). Urbanisation induces early flowering: evidence from *Platanus acerifolia* and *Prunus cerasus*. *International Journal of Biometeorology*, 53, 287–298.
- Mondejar, J.P. & Tongco, A.F. (2019). Near infrared band of Landsat 8 as water index: a case study around Cordova and Lapu-Lapu City, Cebu, Philippines. *Sustainable Environment Research*, 29.
- Moriuchi, E. & Basil, M. (2019). The Sustainability of Ohanami Cherry Blossom Festivals as a Cultural Icon. *Sustainability*, 11, 1820.

- Morrison, M. & Morgan, M.S. (1999). *Models as mediating instruments*. Cambridge University Press, pp. 10–37.
- Newey, S., Davidson, P., Nazir, S., Fairhurst, G., Verdicchio, F., Irvine, R.J., et al. (2015). *Limitations of recreational camera traps for wildlife management and conservation research: A practitioner's perspective*. *Ambio*, 44, 624–635.
- Ohashi, Y., Kawakami, H., Shigeta, Y., Ikeda, H. & Yamamoto, N. (2011). *The phenology of cherry blossom (*Prunus yedoensis* "Somei-yoshino") and the geographic features contributing to its flowering*. *International Journal of Biometeorology*, 56, 903–914.
- 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*.
- Primack, R.B., Higuchi, H. & Miller-Rushing, A.J. (2009). *The impact of climate change on cherry trees and other species in Japan*. *Biological Conservation*, 142, 1943–1949.
- Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., et al. (2017). *Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure*. *Ecography*, 40, 913–929.
- Rousseeuw, P.J. (1987). *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*. *Journal of Computational and Applied Mathematics*, 20, 53–65.
- Roy, D.P., Boschetti, L. & Trigg, S.N. (2006). *Remote Sensing of Fire Severity: Assessing the Performance of the Normalized Burn Ratio*. *IEEE Geoscience and Remote Sensing Letters*, 3, 112–116.
- Sakurai, R., Jacobson, S.K., Kobori, H., Primack, R., Oka, K., Komatsu, N., et al. (2011). *Culture and climate change: Japanese cherry blossom festivals and stakeholders' knowledge and attitudes about global climate change*. *Biological Conservation*, 144, 654–658.
- Shi, P., Chen, Z., Reddy, G.V.P., Hui, C., Huang, J. & Xiao, M. (2017). *Timing of cherry tree blooming: Contrasting effects of rising winter low temperatures and early spring temperatures*. *Agricultural and Forest Meteorology*, 240-241, 78–89.
- Søgaard, A., Ebert, S., Bastings, J. & Filippova, K. (2021). *We need to talk about random splits*. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.
- 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*.

A. Instructor notes

- Thorndike, R.L. (1953). Who belongs in the family? *Psychometrika*, 18, 267–276.
- Tuia, D., Kellenberger, B., Beery, S., Costelloe, B.R., Zuffi, S., Risso, B., et al. (2022). Perspectives in machine learning for wildlife conservation. *Nature Communications*, 13, 792.
- Unwin, A. & Kleinman, K. (2021). The Iris Data Set: In Search of the Source of *Virginica*. *Significance*, 18, 26–29.
- Valavi, R., Elith, J., Lahoz-Monfort, J.J. & Guillera-Arroita, G. (2018). block CV : An r package for generating spatially or environmentally separated folds for k -fold cross-validation of species distribution models. *Methods in Ecology and Evolution*, 10, 225–232.
- Vermote, E., Justice, C., Claverie, M. & Franch, B. (2016). Preliminary analysis of the performance of the Landsat 8/OLI land surface reflectance product. *Remote Sensing of Environment*, 185, 46–56.
- Watt, J., Borhani, R. & Katsaggelos, A. (2020). Machine learning refined.
- Whittaker, R.H. (1962). Classification of natural communities. *The Botanical Review*, 28, 1–239.
- Yau, N. (2015). Visualize this.
- Zeng, X. & Martinez, T.R. (2000). Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, 12, 1–12.
- Zheng, Y., Zhou, Q., He, Y., Wang, C., Wang, X. & Wang, H. (2021). An Optimized Approach for Extracting Urban Land Based on Log-Transformed DMSP-OLS Nighttime Light, NDVI, and NDWI. *Remote Sensing*, 13, 766.

Index

Actionable models, 28
Class imbalance, 78
Confusion table, 78
 Coin-flip classifier, 81
 Constant classifier, 81
 Matthew's Correlation Coefficient, 87
 Negative Predictive Value, 87
 No-skill classifier, 80, 86
 Positive Predictive Value, 87
Cross-validation
 Holdout, 62
 holdout, 45
 k-fold, 63
Leave-one-out, 63
Leave-p-out, 62

Monte-Carlo, 65, 86
testing set, 59
training set, 57
validation set, 58
Data preparation, 29, 45, 68
Decision boundary, 91
Distances
 Euclidean, 23, 41
Feature selection, 76
Gradient descent, 39
Hyper-parameters, 21
Iteration, 24, 44
Learning rate, 45
Linear regression, 39, 65
Loss functions, 41

Mean squared error, 41
Root mean squared error, 47
Model performance, 38, 68
 Paradox of accuracy, 82
Normalized difference, 20
Optimality, 24, 44
 global, 24
 local, 24
Over-fitting, 46, 76
Regularization, 44, 52
Separability, 77, 89
Tolerance, 47
Variance, 23