Machine Learning Guidelines for Natural Resource Management Practitioners

Shih-Ni Prim and Natalie Nelson

2024-05-14

Contents

Chapter 1

Motivation

As machine learning (ML) has become a powerful tool, it is noted by some that ML has not been widely used in environmental studies. Some might prefer process based models, but they can be costly and ML provides a large variety of tools that should be explored. This booklet is meant to provide a concise guide for natural resource management practitioners. This book serves as a staring point rather than a comprehensive resource, so that practitioners can have a basic understanding of how ML works and how to utilize it to analyze data and answer research questions. When appropriate, we provide case studies and R code as well as other online resources to help the readers on the journey of gaining one powerful tool that is becoming omnipresent in the research world.

Chapter 2

Introduction

What is machine learning (ML)? Essentially, machine learning teaches computer models to look for patterns or make predictions. You can think of machine learning models as finding underlying formulas from which the data come. To solve for such formulas, many, many mathematical calculations are involved. As we human beings are prone to mistakes for repetitive task, a more effective way to work with data is to identify a framework and give the framework and data to a computer model. The computer is best at repeating meticulous calculations to find a best guess based on our believes of the system and the data we observed.

Even though ML models are commonly considered black boxes (or some might say opaque boxes), they are not necessarily so. Many ML methods lend easily to interpretation. Random Forests, for example, is a powerful method that can offer a glimpse into what the important variables are, even if the straightforward explanation that can come from a linear model is unavailable. Below we introduce the two main branches of machine learning.

2.1   Supervised Learning

Simply put, supervised learning is when there are true answers for the model. For example, if we want to use environmental traits (temperature, precipitation, chemical composition) to predict algal growth. In a dataset that records such activities, the observed growths are the "true" answers. The model can then use the predictors to make predictions, and the comparison between the predictions and the truth can measure how well the model performs.

2.2 Unsupervised Learning

Unsupervised learning, on the other hand, performs tasks that do not have correct/true answers. For example, for a group of chemicals, some might have more similar traits than the others. Unsupervised learning can perform clustering to find such groupings. This task is considered unsupervised, because the groupings might depend on the context. One could imagine that clustering can be performed first, and the groupings are used as the predictors instead of the individual chemicals. This way the dimension can be reduced, assuming that the number of groups is smaller than the number of chemicals.

2.3 Online Resources

If you feel overwhelmed about the differences in ML models, you are not alone! The sheer number of ML models attests to the power and versatility of machine learning. Search *machine learning models cheat sheet* and you will find some concise resources out there. Here are two that provide useful information.

- Which machine learning algorithm should I use by SAS blogs.
- Machine Learning Cheat Sheet by Datacamp.

2.4 Further Reading

If you would like to learn more, the book *An Introduction to Statistical Learning* by James et al. (2013) is a great resource! This book provides a wealth of information about machine learning models; further, the authors assume that the readers are mainly interested in applying, rather than studying, ML models. The four premises offered by the authors, listed on pp 8-9 of the version about R, demonstrate the practical focus of the book:

1. Many statistical learning methods are relevant and useful in a wide range of academic and non-academic disciplines, beyond just the statistical sciences.
2. Statistical learning should not be viewed as a series of black boxes.
3. While it is important to know what job is performed by each cog, it is not necessary to have the skills to construct the machine inside the box!
4. We presume that the reader is interested in applying statistical learning methods to real-world problems.

There are free PDF versions with applications in R and Python here. Be sure to check it out if you are interested in learning more about machine learning models!

Chapter 3

Data

When you have data, don't feel so rushed to jump into data analysis yet. Some steps can help you know your data better and, more than often, avoid problems down the road. Here we demonstrate typical steps for exploring data.

3.1 Data Exploratoary Analysis

After you read in data, do some checks. Below we use the embedded `mtcars` as an example for illustration.

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----------------------- tidyverse 2.0.0 --
## v dplyr     1.1.4      v readr     2.1.4
## v forcats   1.0.0      v stringr   1.5.1
## v ggplot2   3.4.4      v tibble    3.2.1
## v lubridate 1.9.3      v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```r
data("mtcars")
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
```

```
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

As seen above, the command `str` allows you to see the variables and their types. For this dataset, all variables are numerical. If some are categorical but they should be numerical, make sure you transform them into the right type of data. (Sometimes numbers can be saved as characters, and the analysis would not be correct if the datatype remains as character.)

Next, try to make some plots–typically histograms for numerical variables and barplots for categorical variables.
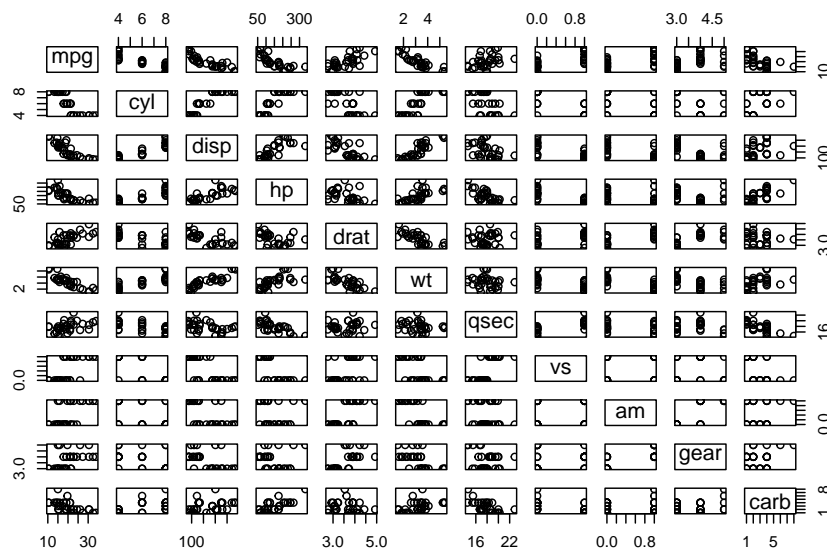
```r
par(mfrow = c(3,3), mar = c(2,2,2,2))
for (i in 1:ncol(mtcars)){
  hist(mtcars[,i], main = paste0("Histogram of ", colnames(mtcars)[i]))
}
```

You can also look at the paired plots to see if two variables are too perfectly correlated, which could cause problems in regression models.

```r
pairs(mtcars)
```

Next, take a look at the summary statistics.

```r
for (i in 1:ncol(mtcars)){
  print(paste0("***** Summaries of ", colnames(mtcars)[i]," *****"))
  print(summary(mtcars[,i]))
}
```

```
## [1] "***** Summaries of mpg *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
## [1] "***** Summaries of cyl *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   4.000   6.000   6.188   8.000   8.000
## [1] "***** Summaries of disp *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    71.1   120.8   196.3   230.7   326.0   472.0
## [1] "***** Summaries of hp *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    52.0    96.5   123.0   146.7   180.0   335.0
## [1] "***** Summaries of drat *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.760   3.080   3.695   3.597   3.920   4.930
## [1] "***** Summaries of wt *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.513   2.581   3.325   3.217   3.610   5.424
## [1] "***** Summaries of qsec *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.50   16.89   17.71   17.85   18.90   22.90
## [1] "***** Summaries of vs *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4375  1.0000  1.0000
## [1] "***** Summaries of am *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4062  1.0000  1.0000
## [1] "***** Summaries of gear *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   4.000   3.688   4.000   5.000
## [1] "***** Summaries of carb *****"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   2.812   4.000   8.000
```

The summaries can show, for example, ranges and means of the variables. They can give you a better understanding of where the values are and whether there might be data entry errors.

3.2 Data Requirement

It is difficult to talk about data requirement in a general sense. A few considerations are offered here. The data should be appropriate to the research questions. For example, if you want to know causal relationships, clinical trial data is the gold standard. Further, make sure that the sample size is large enough for what you want to study. If you would like to know the behavior of mice, repeatedly recording the behavior of one, or even five, mice will not be sufficient. Further, consistency in measurement should be consistent. Any major changes in the data collection methods should be taken into consideration. Sometimes adjustment can be made. The best practice is to consider how the data is collected and ensure that there is sufficient information in the data to answer the research questions.

3.3 Strategies for Missing Data

3.3.1 Removing observations

If you have a large dataset and the percentage of missing data is small, you can simply ignore those observations. Note that you should check whether the pattern of missingness is "missing at random," meaning that the reason why certain observations are missing is pure chance. If, for example, some observations are missing because the pollution level is too high or too low and the detection is not ideal, then ignoring these observations could result in biased conclusions.

3.3.2 Imputation

Imputation is a commonly used method for missing data. If linearity is plausible, you can use linear imputation. If some response variables are missing, some Bayesian method can fill in the values using posterior draws. (In other words, in the process of running the analysis, the algorithm can predict what the response variable should be according to the assumed model and fill it in.) Typically imputation is done multiple times and the mean imputated values are used.

Chapter 4

Evaluation

4.1 Training vs Testing

We should first address the concepts of training and testing. Instead of using the entire data, it is common to split the dataset into a training set and a testing set. Typically people use percentages such as 70% vs 30% or 80% vs 20%. The idea is to use only, say, 80% of the data for training, which means to run the select model on this part of data. Then you have a model with some coefficients (you can think of them as slopes) and test such results on the testing set to see how well the model performs. Since the part of the data for testing is not used in training, there is no double dipping. Sometimes people split the dataset into three sets: training, testing, and validation. So the trained models are used on the testing set for model selection. Then the validation set is used only once at the end to gauge how the model should perform if there's new data.

4.2 Metrics

4.2.1 Numerical Responses

The most common metrics for numerical responses include mean squared errors (MSE), bias, and R^2 (or adjusted R^2). The way to calculate MSE is:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}_i - y_i \right)^2,$$

where $\hat{y}_i$ is the predicted outcome of the ith observation and y_i is the ith observed outcome. The metric shows, on average, the squared distance between

predictions and true responses. Bias is calculated as

$$Bias = \frac{1}{n} \sum_{i=1}^{n} \left(\widehat{y}_i - y \right).$$

The two metrics seem similar, but they reveal different aspects of model performance. In general, bias shows if the predictions are centered around the truth, while the MSE shows whether the predictions are centered around the truth *and* how far the predictions are from the truth. If the bias is zero, it means that the mean prediction goes towards the truth as the sample size increases to infinity. This is naturally a desired property, but you might be surprised that sometimes we are willing to accept some bias to reduce the variance.

One important formula,

$$MSE = Bias^2 + Variance,$$

where

$$Variance = \frac{1}{n} \sum_{i=1}^{n} \left(\widehat{y}_i - E(\widehat{y}) \right)^2$$

and

$$E(\widehat{y}) = \frac{1}{n} \sum_{i=1}^{n} \widehat{y}_i.$$

We skip the derivation here, but the interested readers can find the derivation readily online. The formula tells us that, if the MSE is large but the bias is small, this means the model has a large variance. This is to say, if you fit the model multiple times, it has quite different results every time. These formulas might seem too technical, but the message is important. There is a trade off between the bias and variance; in other words, there is no free lunch. A balance between bias and variance is where many methods strive to achieve.

4.2.2   Categorical Responses

When the responses are categorical, a common way to measure the performance is a confusion matrix, which shows the numbers of predictions that are correct or incorrect. See some examples in Workflow Demonstration. Other metrics include sensitivity, specificity, precision, recall, and other metrics that try to combine these metrics to present a one-number summary.

4.3   Overfitting

Overfitting means that the model can make good predictions with the available data, but it cannot generalize well on new data. It is like if a student memorizes

all the past exam questions without really studying the materials, then, on the exam, the student can do really well on questions coming from past exams but might do poorly on questions never seen before. Splitting data into training and testing is one strategy to avoid overfitting. Certain ML methods are less likely to overfit, such as neural networks. And there are some more strategies, such as early stopping, that can be used. Our point here is to be aware of this potential pitfall of machine learning methods.

4.4 Cross Validation

One very standard way of evaluation is k-fold cross validation, commonly with $k = 5$ or $k = 10$. The idea is simple. Divide the data into k groups. Each time, choose $k - 1$ groups for training, fit the model on the last group, which is the test data, and calculate the desired metrics, such as MSE.

This way, although less data is used for training, the metrics are more accurate, because now we are not using the same data points for training and testing. Using metrics from cross validation for model selection can ensure that your model does not overfit.
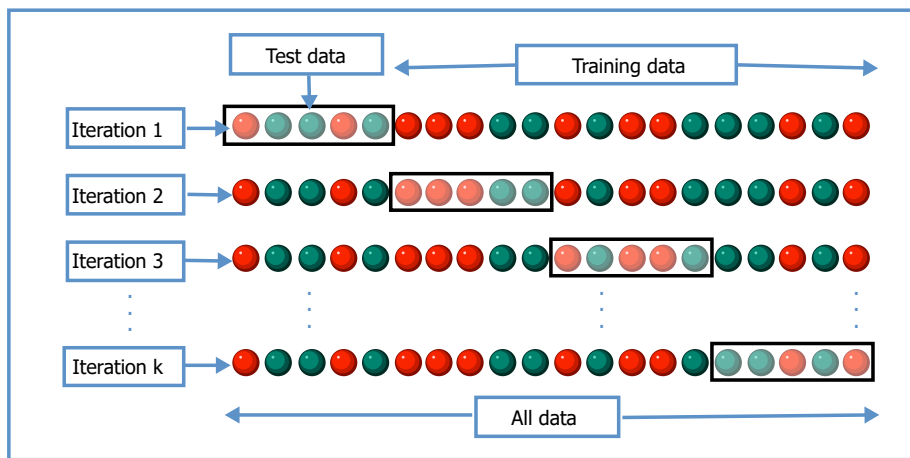

Figure 4.1: Image Source: https://en.wikipedia.org/wiki/Cross-validation_ (statistics)

Chapter 5

Machine Learning Methods

Here we discuss some machine learning methods you might find useful. Although linear regression models are commonly used, we skip it because it is well understood and many online resources can be easily located.

5.1 Tree-based Methods

Tree-based methods are popular for their ease for interpretation. Essentially, the algorithm tries to split the data into subsets by finding the most significant (or consequential) variables in a way that reduces the prediction error the most. For example, in the graph below, the variable that helps reduce prediction error at the first level is gender. In other words, the object being male or female can greatly help the algorithm predict the survival of passengers on Titanic.

Decision trees are easy to use but not very accurate for complex data; however, we can go beyond one tree. If we use multiple trees and, for instance, take the average of the predictions of all the trees, we can improve the performance. This is the idea of ensemble methods. One of the ensemble methods, Random Forests, is commonly used, including in natural resources management, and quite accurate for many kinds of data. So next we'll dive into this one method.

5.1.1 Random Forests

For Random Forests, the algorithm randomly selects a smaller number, say m, of predictors from the total number, say p, each time a tree is being fit. This process is repeated many times, for example, 500 times. The average (for continuous responses) or the majority votes (for discrete responses) of the predictions from the individual trees are used as final predictions. By randomly
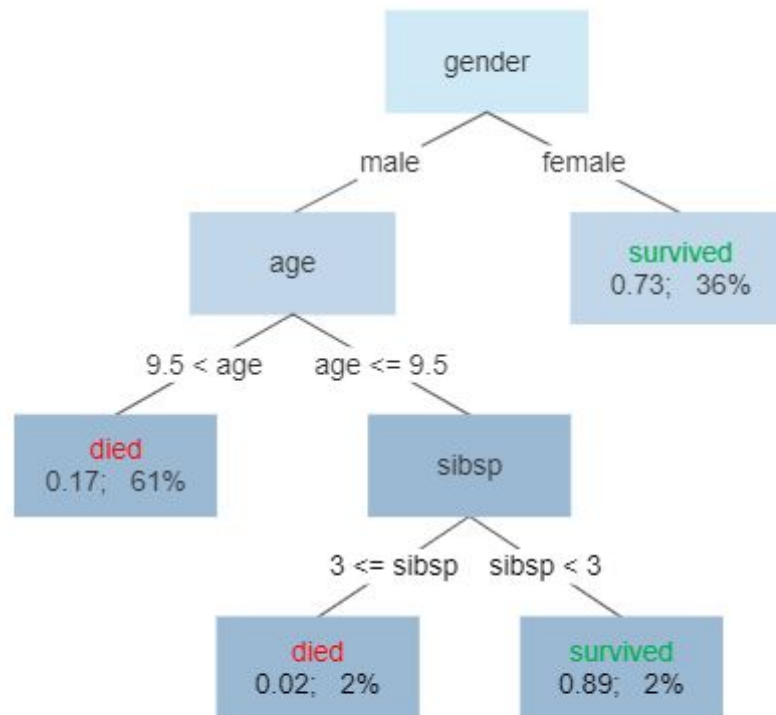
Figure 5.1: By Gilgoldm - Own work, CC BY-SA 4.0, https://commons. wikimedia.org/w/index.php?curid=90405437

choosing a subset of predictors each time, random forests can avoid relying on the same variables every time and prevent overfitting.

Even though coefficients of each predictors are not available for random forests, you can easily get the important variables and at least know which variables are more important for predicting the response. You can also use Random Forests as a tool for variable selection and then use a different method to fit the model on a subset of predictors chosen by Random Forests.

See Workflow Demonstration for examples of Random Forests models.

5.1.2 Other Ensemble Tree Methods

Other ensemble tree methods include boosting and bagging. If you are interested, take a look at Chapter 8 (Tree-Based Methods) of *An Introduction to Statistical Learning* by James et al. (2013).

5.2 Neural Network

Neural network has been enjoying its popularity. A neural network model is composed of an input layer, some hidden layers, and an output layer, as seen below. There is only one hidden layer below. When there are enough hidden layers, the model is then called deep learning.

So the input layer is the observed predictors, and each of the hidden layers has nodes that contain coefficients. You can think of these coefficients similar to those for linear models, except they do not have linear relationships with the observed predictors. These values are derived from finding the best fit between data and some kind of cost function; essentially, these values are meant to minimize the differences between observed outcome and predictions. When the values from one hidden layer are passed onto the next layer, a nonlinear activation function called ReLU is commonly used. If the task is classification, the layer right before output is often some kind of algorithm to turn the values into predicted class. While neural network is a powerful ML model, the values in the nodes in the hidden layers are not interpretable. So this indeed is a black-box method. For interested readers, check out Chapter 10 Deep Learning in James et al. (2013) (*An Introduction to Statistical Learning*).

5.3 Gaussian Process (GP)

Another powerful ML method that could be useful for natural resource management type of research is Gaussian Process. This method, commonly called kriging in spatial statistics, is non-parametric, which sounds like there are no
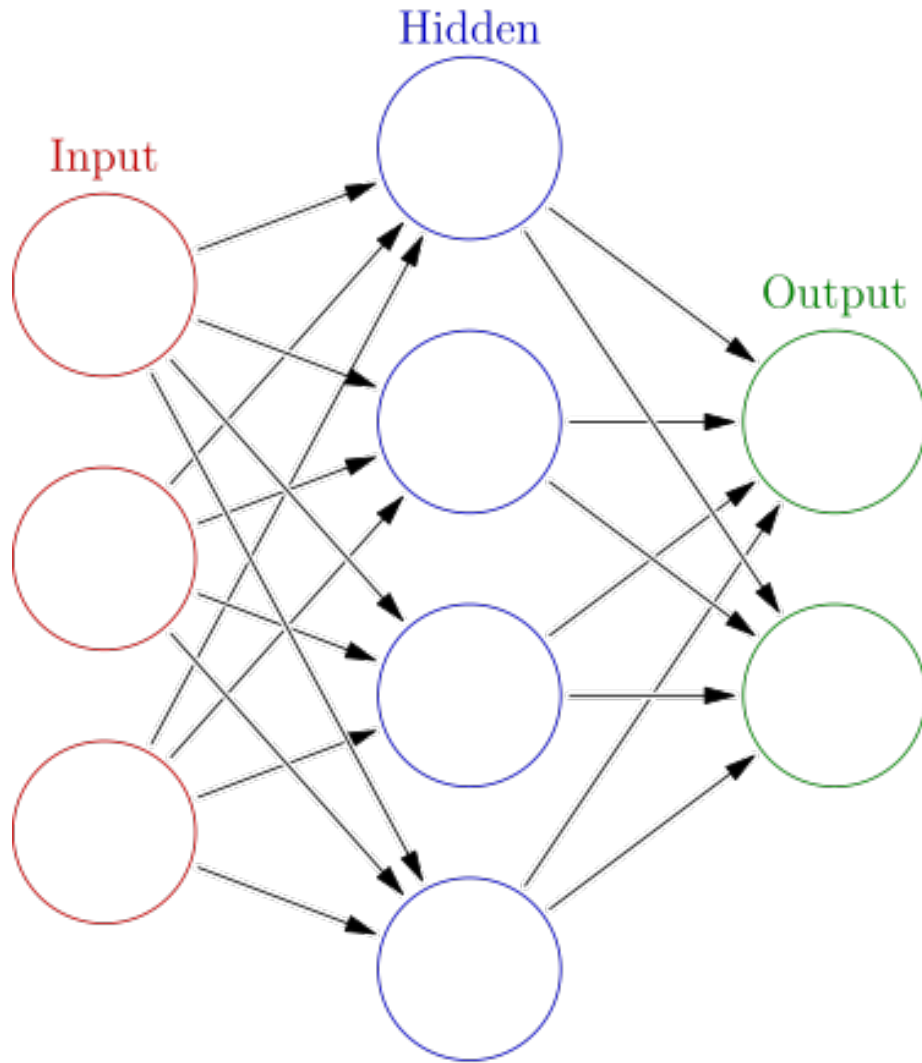
Figure 5.2: By Glosser.ca - Own work, Derivative of File: Artificial neural network.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24913461

parameters in the model but in fact means there can be an infinite number of parameters. (Of course in any real model we'll have finite parameters, but that's the idea.) The fact that the parameters can increase towards infinity tells us that this is a flexible model. Essentially, for a GP model, and here I quote from Chapter 5 of Gramacy's *Surrogates*, "any finite collection of realizations (i.e. n observations) is modeled as having a multivariate normal (MVN) distribution." In other words, the model tries to learn information from similar items, judging from some kind of distance from each other. MVN has some great properties. And, when there is a new data point, all the observed data points can be used to make prediction for it.

GP is widely used in the machine learning, computer experiments, and spatial statistics communities. There are ready-made R packages, such as `GpGp` and `deepgp`. It's also easy to use in Python. Chapter 5 ("Gaussian Process Regression") of the book *Surrogates* by Gramacy (2020) is a great resource if you are interested in learning more about Gaussian process! If you would like to see some theory of GP, check out another classic book *Gaussian Processes for Machine learning* by Rasmussen and Williams (2006).

Chapter 6

Workflow Demonstration

In this section, we use a dataset to exemplify a typical workflow for constructing ML models. We skip exploratory data analysis, since we have already addressed that aspect in Data. The dataset contains wine quality and traits, and our goal is to predict the quality of wine using the traits. The dataset can be found here.

```r
knitr::opts_chunk$set(echo = TRUE, eval = TRUE, cache = TRUE)
library(tree)
library(tidyverse)
library(caret)
library(rattle)
library(randomForest)
```

6.1 Prepare Data

We first read in the data and rename the variables, so coding is easier.

```r
wine <- read_delim("materials/winequality-red.csv", delim = ';')
```

```
## Rows: 1599 Columns: 12
## -- Column specification -----------------------------------------------
## Delimiter: ";"
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# shorten variable names
fa <- wine$`fixed acidity`
va <- as.numeric(wine$`volatile acidity`)
ca <- as.numeric(wine$`citric acid`)
rs <- wine$`residual sugar`
ch <- as.numeric(wine$chlorides)
fsd <- wine$`free sulfur dioxide`
tsd <- wine$`total sulfur dioxide`
den <- as.numeric(wine$density)
ph <- wine$pH
sul <- as.numeric(wine$sulphates)
al <- wine$alcohol
qual <- wine$quality
winez <- data.frame(fa, va, ca, rs, ch, fsd, tsd, den, ph, sul, al, qual)
```

To demonstrate how to construct Random Forests models with continuous and discrete outcomes, we also transform the continuous variable, wine quality, our response variable, into two discrete variables. One has two levels: high vs low, and one has three levels: H, M, and L. The thresholds are decided rather arbitrarily, and you can use your domain knowledge to gauge how to set the thresholds.

```r
# collapse qual into 2 labels
winez$qual2 <- as.factor(ifelse(winez$qual < 6, "low", "high"))
# collapse qual into 3 labels
winez$qual3 <- as.factor(ifelse(winez$qual < 5, "L", ifelse(winez$qual < 7, "M", "H")))
table(qual)
```

```
## qual
##   3    4    5    6    7    8
##  10   53  681  638  199   18
```

```r
table(winez$qual2)
```

```
##
## high  low
##  855  744
```

```r
table(winez$qual3)
```

```
##
##    H    L    M
##  217   63 1319
```

Next we randomly separate the dataset into a training set (80% of the rows) and a testing set (20% of the rows). A seed is set so that the result can be reproduced.

```
# separate data into a training set and a test set
set.seed(2)
train <- sample(nrow(winez), nrow(winez)*.8)
winez_train <- winez[train,]
winez_test <- winez[-train,]
```

6.2 Classification Models

Now we construct a Random Forests model with the two-level response. After the model is run, we calculate the predictions using the `predict` function. A plot for the important predictors are provided. We can see that `alcohol`, `sulphates`, and `volatile acidity` are three most important predictors for the outcome variable. A confusion matrix is created, which shows different metrics such as overall accuracy rate, sensitivity, specificity, etc.

```
# random forest
rfGrid <- expand.grid(mtry = 2:8)
# 2-level variable
rf_tree2 <- train(qual2 ~ fa + va + ca + rs + ch + fsd + tsd + den + ph + sul + al, data = winez_
rf2_pred <- predict(rf_tree2, newdata = winez_test)
rfMatrix2 <- table(rf2_pred, winez_test$qual2)
rf2_test <- mean(rf2_pred == winez_test$qual2)
plot(varImp(rf_tree2))
```
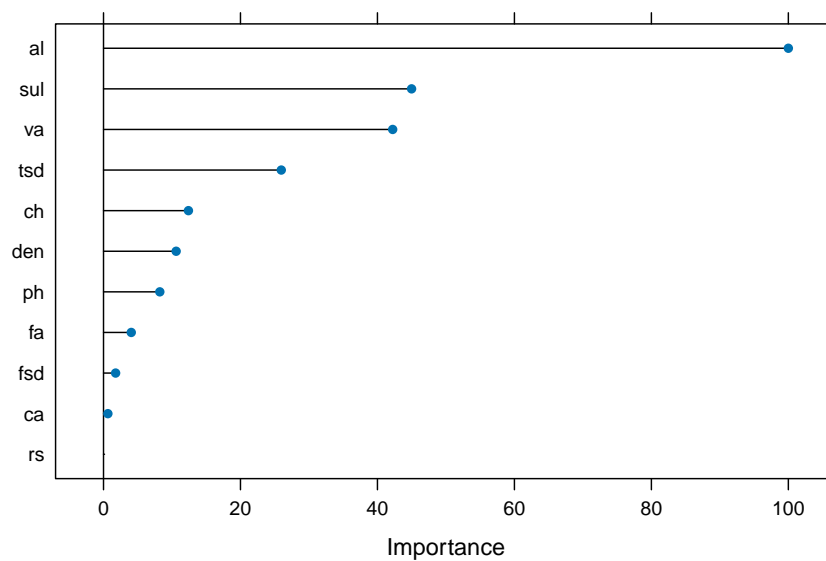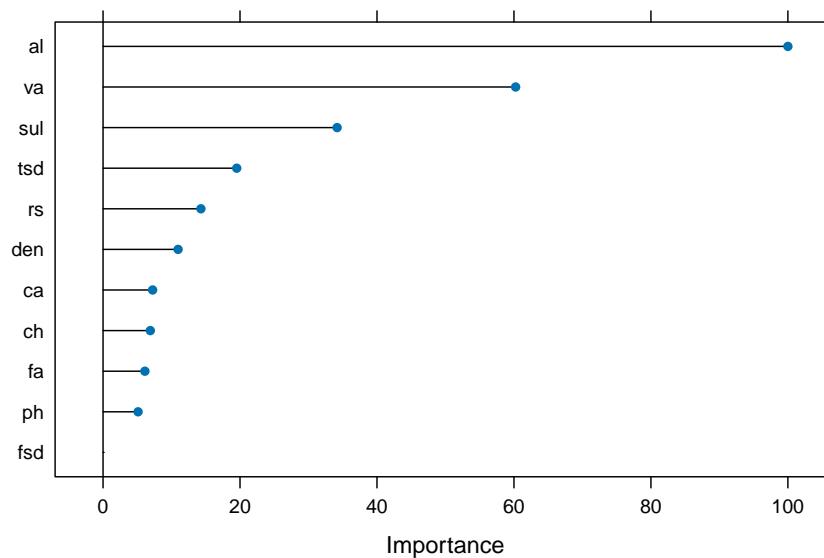
```r
confusionMatrix(rf2_pred, winez_test$qual2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  138  25
##       low    45 112
##
##                Accuracy : 0.7812
##                  95% CI : (0.7319, 0.8253)
##     No Information Rate : 0.5719
##     P-Value [Acc > NIR] : 2.968e-15
##
##                   Kappa : 0.5613
##
##  Mcnemar's Test P-Value : 0.02315
##
##             Sensitivity : 0.7541
##             Specificity : 0.8175
##          Pos Pred Value : 0.8466
##          Neg Pred Value : 0.7134
##              Prevalence : 0.5719
##          Detection Rate : 0.4313
```

```
##       Detection Prevalence : 0.5094
##          Balanced Accuracy : 0.7858
##
##              'Positive' Class : high
##
```

Next we create a model with the three-level outcome variable. Again, important predictors are plotted and a confusion matrix is included below. `alcohol`, `volatile acidity`, and `sulphates` are again the three most important predictors, but the order changed. The overall accuracy is higher here than the two-level model.

```
# 3-level variable
rf_tree3 <- train(qual3 ~ fa + va + ca + rs + ch + fsd + tsd + den + ph + sul + al, data = winez_
rf3_pred <- predict(rf_tree3, newdata = winez_test)
rfMatrix3 <- table(rf3_pred, winez_test$qual3)
rf3_test <- mean(rf3_pred == winez_test$qual3)
plot(varImp(rf_tree3))
```



```
confusionMatrix(rf3_pred, winez_test$qual3)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   H   L   M
##          H  25   0   5
##          L   0   0   1
##          M  18   8 263
##
## Overall Statistics
##
##                   Accuracy : 0.9
##                     95% CI : (0.8618, 0.9306)
##        No Information Rate : 0.8406
##        P-Value [Acc > NIR] : 0.001473
##
##                      Kappa : 0.5617
##
##   Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: H Class: L Class: M
## Sensitivity           0.58140 0.000000   0.9777
## Specificity           0.98195 0.996795   0.4902
## Pos Pred Value         0.83333 0.000000   0.9100
## Neg Pred Value         0.93793 0.974922   0.8065
## Prevalence            0.13437 0.025000   0.8406
## Detection Rate         0.07812 0.000000   0.8219
## Detection Prevalence  0.09375 0.003125   0.9031
## Balanced Accuracy      0.78167 0.498397   0.7339
```

The plot below shows how the accuracy rate changes with the number of randomly selected variables in each tree (denoted as m).

```
# random forest plot: accuracy rates vs number of predictors
rfplot1 <- plot(rf_tree2)
rfplot2 <- plot(rf_tree3)
gridExtra::grid.arrange(rfplot1, rfplot2, nrow = 1, ncol = 2)
```

6.3   Prediction Models

Next we construct Random Forests models with the continuous outcome variable. We first set m as the square root of the number of predictors, as this is commonly recommended. Confusion matrices cannot be provided for continuous responses, but we can calculate MSEs, find important variables, and show
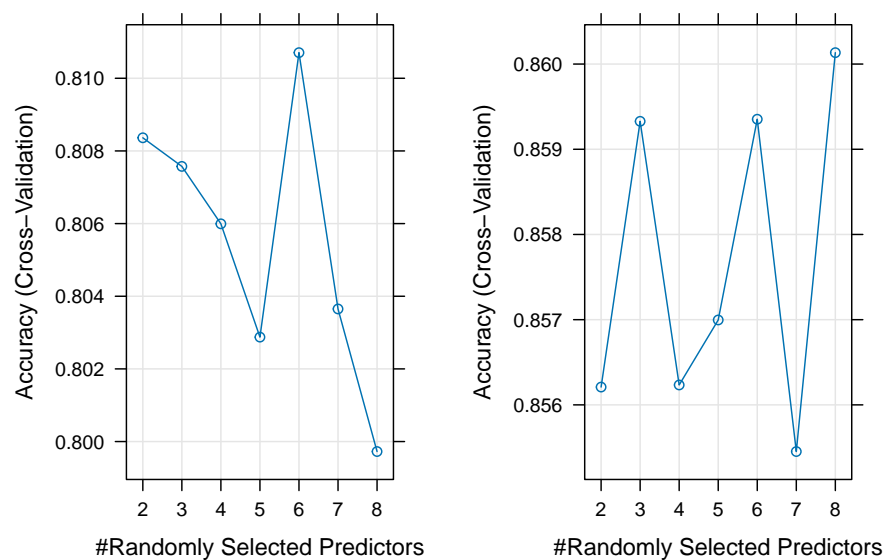
Figure 6.1: Accuracy Rates vs Number of Predictors Used for 2-level Variable (left) and 3-level Variable (right)

how the error decreases with the number of trees constructed. The three most
important variables are again `alcohol`, `sulphates`, and `volatile acidity`.

```r
# randomforests, mtry = sqrt(11)
rf.def.Wine <- randomForest(qual ~ fa + va + ca + rs + ch + fsd + tsd + den + ph + sul
yhat.rf.Wine <- predict(rf.def.Wine, newdata = winez_test)
rf.mtry3.testMSE <- mean((yhat.rf.Wine - winez_test$qual)^2)
varImp(rf.def.Wine)
```

```
##       Overall
## fa   22.11429
## va   37.04878
## ca   23.74687
## rs   18.55913
## ch   24.96786
## fsd 21.65164
## tsd 32.28444
## den 27.19701
## ph   22.36903
## sul 43.52900
## al   54.53284
```

```r
plot(rf.def.Wine)
```
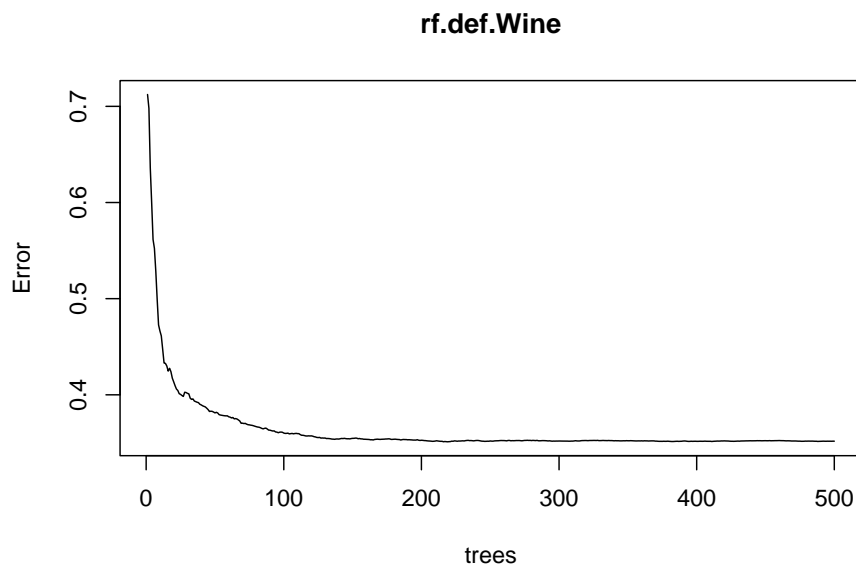
We try several different numbers for the number of variable included in each tree.
`alcohol`, `sulphates`, and `volatile acidity` remain the three most important
predictors. Two plots are provided to show how the error decreases with the
number of trees.

```
# mtry = 4
rf.def.Wine.m4 <- randomForest(qual ~ fa + va + ca + rs + ch + fsd + tsd + den + ph + sul + al, 
rf.pred.m4 <- predict(rf.def.Wine.m4, newdata = winez_test)
rf.mtry4.testMSE <- mean((rf.pred.m4 - winez_test$qual)^2)
varImp(rf.def.Wine.m4)
```

```
##       Overall
## fa   22.78226
## va   39.40868
## ca   22.22551
## rs   16.27685
## ch   25.12775
## fsd 21.58168
## tsd 30.36789
## den 27.20655
## ph   20.22223
## sul 47.59070
## al   60.54909
```

```
# mtry = 7
rf.def.Wine.m7 <- randomForest(qual ~ fa + va + ca + rs + ch + fsd + tsd + den + ph + sul + al, 
rf.pred.m7 <- predict(rf.def.Wine.m7, newdata = winez_test)
rf.mtry7.testMSE <- mean((rf.pred.m7 - winez_test$qual)^2)
varImp(rf.def.Wine.m7)
```

```
##       Overall
## fa   22.31197
## va   43.85162
## ca   22.34412
## rs   15.79913
## ch   26.89172
## fsd 22.86225
## tsd 35.73985
## den 28.37794
## ph   24.39275
## sul 55.85386
## al   71.99840
```

```r
par(mfrow = c(1, 2))
plot(rf.def.Wine.m4)
plot(rf.def.Wine.m7)
```
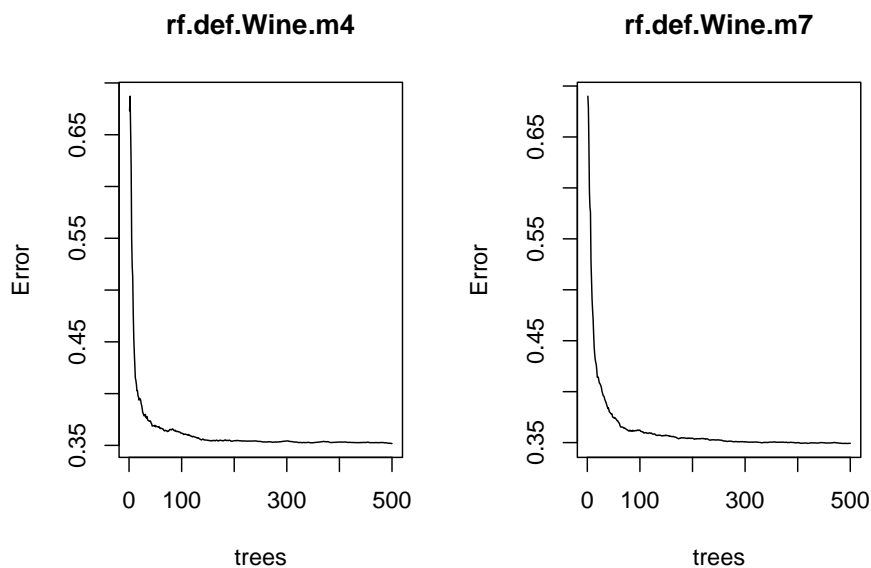

Lastly, we put the MSEs from these models together. The differences are very small, but the model that uses $m = \sqrt{p}$ has a slightly lower MSE.

```r
res <- data.frame(rf.mtry3.testMSE, rf.mtry4.testMSE, rf.mtry7.testMSE)
colnames(res) <- c("squre root of p", "4", "7")
rownames(res) <- c("MSE")
knitr::kable(res)
```

	squre root of p	4	7
MSE	0.289515	0.2907887	0.292777

Chapter 7

Case Studies

Since ML models are relatively easy to construct, they can at least be used to compare results or explore areas in which not much is known, so hypotheses can be made to help guide the directions of process-based models. Here we provide overviews of three case studies to demonstrate how ML models can be beneficial for environmental and natural resource management studies.

7.1 Case Study 1: Machine learning approach for modeling daily pluvial flood dynamics in agricultural landscapes

7.1.1 Takeaway

With the remote sensed images as important data sources, the researchers had to do a lot of pre-processing. While pre-processing images might seem time consuming, once the workflow is established, studying flood in other areas becomes easier, as long as images are available. This is one of the strengths of ML models–the process can be at least partially automated, and data is more easily obtained than if measuring at each site is required.

7.1.2 Summary

In this study, Fidan et al. (2023) build a Random Forests model using gridded rainfall data, derived from remotely sensed imagery, from the 2016 hurricane Matthew's impact in Kinston, North Carolina. Besides the proprietary images, the researchers also included readily available geospatial landscape traits for the models. With the objectives of (1) develop a pluvial flood dataset using

imagery, (2) test the performance of Random Forests models for the modeling flood in "low-lying and flat agricultural terrain," (3) find important predictors, and (4) "generate pluvial flood time series."

The outcome variable is binary in terms of flooding. Each pixel is treated as one data point. Non-flooded areas were covered by 5,875,480 pixels, and flooded areas by 678,010 pixels. Since there were many more non-flooded pixels than flooded pixels, overall accuracy alone would not be a good performance metric, since, if the model always predicts that a pixel is not flooded, the accuracy rate would be high. The researchers chose other metrics such as precision, recall, specificity, and F1 scores to reflect model performance in different scenarios. The model achieved an overall accuracy of 0.97 and an F1 score of 0.69. Considering that the no information rate (if the model predicts non-flooded all the time) is around 88.5%, this performance seems satisfactory. They found the important variables to be population density, distance to the nearest river, height above nearest drainage, and distance to the nearest road.

The full paper can be accessed here.

7.2   Case Study 2: Short-term forecasting of fecal coliforms in shellfish growing waters

7.2.1   Takeaway

This study serves as a nice case study because (1) it demonstrates that Random Forests does well with nonlinear relationships between the predictors and outcomes, (2) the performance is at least as good as other studies, (3) the findings are consistent with what researchers have known. These reasons help us establish trust in Random Forests models. Furthermore, Random Forests models are easy and cheap ton construct. The R package `caret` and function `rf`, used by Chazal et al. (2024), is easy to use and quick to run, if the number of trees are reasonably chosen. The important variables, readily provided by `caret::rf`, help greatly with interpretation.

7.2.2   Summary

This study's goal was to predict near-term (1-3 days) fecal contamination in coastal shellfish growing waters. For each of the five management areas, Chazal et al. (2024) constructed five Random Forests models to (1) use watershed characteristics as well as antecedent hydrologic and meteorologic observations to predict the level of fecal coliform (FC), (2) test whether forecasted rainfall can be useful for predictions, and (3) find important variables. Let's focus on goals (1) and (3) for this case study.

For their first goal, depending on the management areas, the R^2 value is between 0.40 and 0.74. According to Chazal et al. (2024), this performance is similar to that of previous studies. The other settings are typical of ML studies. They split the data into 80% training and 20% testing and used the variance inflation factor (VIF) to remove variables that are collinear, which means some variables are correlated with each other. For their third goal, they found (1) antecedent rainfall, (2) river stage threshold, and (3) wind are high on the importance for prediction. These findings are consistent with the current understanding. In short, this study demonstrates the reliability and ease for construction and interpretation of random forest models.

The full paper can be accessed here.

7.3 Case Study 3: In-season Sweetpotato Yield Forecasting using Multitemporal Remote Sensing Environmental Observations and Machine Learning

7.3.1 Takeaway

Although its results are considered moderate, this study demonstrates how ML models can be used to help with decision making and planning. As Carbajal-Carrasco et al. (2024) state, "Even though ML models are unable to explain underlying processes, they can surpass the predictive accuracy of process-based models, making ML algorithms particularly useful for yield forecasting at scales that are often computationally prohibitive for process-based models." Using remote sensing data and considering a multitude of predictor sets, this study is able to test out how to find the best combination of predictors, the best aggregation periods, and the optimal timing of constructing models. It is promising that they find that the end of the mid-season is an optimal time for constructing models, leaving four to eight weeks for planning. One can assume that with further examination and inclusion of other predictors, some models can give reliable forecasting in advance. Such models are highly applicable in other fields to help with policy making, process planning, and even disaster preparedness.

7.3.2 Summary

Using stationary and temporal predictors from multi-temporal remote sensing data to make early yield forecasting, Carbajal-Carrasco et al. (2024) use four ML methods–Random Forest Regression (RFR), Artificial Neural Networks (ANN), Support Vector Machine (SVM), and Extreme Gradient Boosting (XGB)–to construct sweetpotato yield models at the county level in and around the Coastal

Plain in North Carolina from 2008 to 2022. Besides the primary goal to predict yield, the authors also aim to (1) identify key predictors through variable selection, (2) implement the four ML methods, (3) determine the optimal aggregation periods for temporal variables, and (4) evaluate how early in the season the models can reliably predict end-of-season yields. The predictors considered include topography and soil traits (elevation, slope, aspect, sand, clay, pH, cation exchange capacity, bulk density, nitrogen, soil organic carbon), weather (maximum and minimum temperature around planting, precipitation) and vegetation greenness (NDVI). They considered 16- and 32-days as the time frames to aggregate data as well as comparing model performance of early, mid, and late models. They used the Boruta method for variable selection.

The authors chose the Random Forest Regression algorithm with 16-day composite predictors using early- to mid-growing season as the best model. The R^2 value is 0.44 and RMSE (root mean squared error) as $3.53 t.ha^{-1}$. The authors describe the two above metrics as acceptable for forecasting. Elevation was found to be the most important predictor. They also conclude that the best time to run the model is before the end of the mid-season, approximately four to eight weeks before harvest. The authors cite the small sample size (95 records from 17 counties) as a major limitation and propose other predictors (e.g. solar-induced chlorophyll fluorescence) and ML models (e.g. Deep Learning) that could improve the performance.

The full paper can be accessed here.

Chapter 8

Ethical Considerations

Ethics might sound heavy, but this section is mainly about ensuring that the analysis is carried out in a responsible and reproducible way. These concerns are not unique to ML models, but the complexity of ML methods makes it even more important to think about these issues.

8.1 Rigor

As researchers, we are responsible for our findings. Constructing ML models is similar to using statistical methods. If one goes through all combinations of predictors and fits a method many, many times, one of the models are bound to have a "statistical significant" result. It is thus important to report all models constructed and use multiple testing adjustment when applicable (see Chapter 13 "Multiple Testing" in *An Introduction to Statistical Learning* by James et al. (2013)).

8.2 Reproducibility

To allow for others to reproduce your work, it is important to provide enough details in terms of methods, data processing, code implementation, etc. It is also encouraged to have all the code and data available in a repository. If parts or all of the data should not be shared publicly, it helps to provide a simulated data set.

Another aspect of reproducibility might be surprising. Try to use programming scripts rather than drag-and-drop software. Starting from reading in the data, write a script to read, clean, and wrangle with the data. The scripts can preserve all the steps of data analysis. In the future, when you cannot remember how

you changed the data or how you arrived at a p-value, the scripts can show all the details, but drag-and-drop software will not leave any trace.

Note that the best practice is to read in the data into the programming environment and make any data cleaning and merging in the script. You can save the cleaned version in a different file, but don't change the original file with the raw data.

8.3   Decision making

Since research related to environmental sciences and natural resources could likely affect decision making, we will now address some topics.

8.3.1   Uncertainty qualification

While all supervised ML models can provide point estimates, not all can quantify uncertainty. It is, however, important to show how confident the model is about the estimates. If the conclusion of the study could affect an important policy change, it is crucial to present a full picture of the findings, which include uncertainty quantification. It is wildly different whether the model is 20% or 95% confident about its answer, for instance.

8.3.2   Interpretability

While some models, such as neural networks, are highly efficient, they are more like black boxes and do not lend easily to interpretablity. In the case of neural networks, even if you are able to find all the weights in the hidden layers, there is really no way to interpret them. To ensure that the model arrives at a reasonable conclusion, you might consider using a model that is more interpretable, such as linear regression or tree-based methods. This way, experts with domain knowledge can examine whether the conclusion makes sense. In other words, the findings from ML models can add to researchers' understanding of the field rather than throwing out an answer that is not easily interpreted.

Chapter 9

Appendix

9.1 Do's and Don'ts

Below we provide some tips for what to do and what not to do when you build machine learning models.

9.1.1 Do's

- Do data exploratory analysis before jumping into analysis. Look at summary statistics and make plots to familiarize yourself with the data.
- Document the process with reasonable details, so you or other researchers can reproduce the study if needed.
- Along the same line, if you write script for your code, add comments along the way.
- In the publication, explain what the result entails and the limitations of the model.
- When possible, use interpretable and/or parsimonious models. Think Occam's razor.
- Make data and code available to others if possible.
-

9.1.2 Don'ts

- Don't alter the raw data. Use a script to process data instead.
- Don't over interpret the result. For example, if the samples all come from one geographic area, you might not be able to generalize the result to other areas.

-

Bibliography

Carbajal-Carrasco, M., Jones, D., Williams, C., and Nelson, N. G. (2024). In-season sweetpotato yield forecasting using multitemporal remote sensing environmental observations and machine learning. *ESS Open Archive*.

Chazal, N., Carr, M., Leight, A. K., Saia, S. M., and Nelson, N. G. (2024). Short-term forecasting of fecal coliforms in shellfish growing waters. *Marine Pollution Bulletin*, 200:116053.

Fidan, E., Gray, J., Doll, B., and Nelson, N. G. (2023). Machine learning approach for modeling daily pluvial flood dynamics in agricultural landscapes. *Environmental Modelling and Software*, 167:105758.

Gramacy, R. B. (2020). *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.