A Machine Learning Primer for Natural Resource Management

Natalie Nelson, Shih-Ni Prim, et al.

2024-10-22

Contents

Chapter 1

Welcome

From satellites to field-deployable sensors, the entire agricultural and environmental landscape is increasingly monitored. Advances in sensing allow for a wide range of variables to be measured at unprecedented rates and scales. The large amounts of data produced from agricultural and environmental systems are paving the way for data-intensive methods like machine learning and Artificial Intelligence to support decision-making in natural resources management. But these opportunities also create new educational needs, particularly among applied scientists and engineers who may not have received formal training in "data science" methods (Saia et al., 2021), which is why we made this primer.

1.1 How to use this primer

This primer serves as a succinct guidebook on using machine learning in natural resources management. It is written as a starting point, and not a comprehensive resource.

All readers should start with chapter 2, which defines and describes "machine learning". If you are not interested in developing machine learning models, and instead want to understand how to interpret machine learning models or critique tools that use machine learning, continue reading through chapters 3 and 4. If you are interested in seeing an example workflow for developing a machine learning model, skip to chapter 5.

1.2 Learning how to code

For those of you who are interested in developing machine learning models, the material presented in this primer will help show you where to start. In chapter

5, we include an example using code prepared in R, an open source statistical software environment. To learn more about coding in R, we recommend R for Data Science.

1.3 Acknowledgements

Support for this primer was provided by the U.S. Department of Agriculture National Institute of Food and Agriculture grant number 2019-67021-29936.

1.4 Citation

Authors vary by chapter (2024), A Machine Learning Primer for Natural Resources Management, Accessed online via go.ncsu.edu/mlprimer.

1.5 License

Chapter 2

What is machine learning?

Authors: Natalie G. Nelson, Shih-Ni Prim, Sheila Saia, Khara Grieger, Anders Huseth, Brian Reich

Machine learning methods teach computer models to make predictions from patterns in data (Zhi et al., 2024)[https://doi.org/10.1038/s44221-024-00202-z]. To find patterns in data, many, many mathematical calculations are involved. Computers are needed to develop machine learning models, and advances in computing gave rise to machine learning.

2.1 How is machine learning useful for natural resources management?

Historically, environmental and agricultural systems were primarily modeled with "process-based" or "mechanistic" models, which are models that simulate key system processes based on their underlying physics, chemistry, or biology (Haefner 2005). Models were often used to fill gaps in direct measurements of various environmental and agricultural phenomena. In the past, collecting any type of environmental or agricultural data was grueling, time consuming, and done by people. Today, some variables still require tough and tedious work to measure, but many variables are now readily measured with sensors and other automated instruments. As an example of how process-based models are used, let's consider the flow of water through an open channel. If you were estimating the velocity of water through an open channel using a process-based approach, you may use Manning's Equation, which simulates water velocity as a function of the channel's physical properties like its slope and cross sectional area. To apply Manning's Equation, you would have to take measurements of the channel's dimensions.

However, with machine learning, you do not have to simulate environmental and agricultural variables using underlying processes. Instead, you can create a model that makes predictions from patterns in data. Machine learning models require that you have some measurements of your *response* or *target* variable, which is the variable you are seeking to estimate. In our current example, water velocity in the channel is the target variable. You also need measurements of *predictor* variables, or variables that you will use to predict your response.

As one hypothetical example, a machine learning model of water velocity in an open channel might make predictions based on images instead of physical channel properties. A camera could be situated towards the channel to take time-lapse photos every few minutes. A machine learning model could then be developed to predict water velocity based on the appearance of the water in the images. For a real-world example of this, check out Chapman et al. (2024), Stage and discharge prediction from documentary time-lapse imagery, in *PLoS ONE*.

To create such a machine learning model, water velocity measurements would be needed to create a *training set*, or data used to *train* or develop the machine learning model. The water velocity measurements would be collected at the same time as the images, so the images could then be directly related to water velocity measurements (i.e., when the image looks like this, the water velocity is that). The *predictor* variables would be derived from the images. A machine learning algorithm could be selected to search for patterns between the images – e.g., the shading/color of individual image pixels, relationships between neighboring pixels – and the water velocity measurements. Once these patterns are established, they can be used to estimate water velocity from new images as they are collected. Importantly, the image-based machine learning model knows nothing about the underlying physics controlling water velocity in the channel; it has simply learned that certain patterns in the images correspond to higher or lower water velocities.

Machine learning models can easily be expanded to include many different types of predictor variables. For example, the image-based machine learning model for water velocity could be further built out to include additional predictors like rainfall, irrigation, and time of year. The flexibility of machine learning models allows them to consider many diverse streams of information when learning patterns from which to make predictions. To see additional examples of machine learning models developed for natural resource management applications, see chapter 4.

2.2 Pros and cons

Because they make predictions from correlative patterns between the predictor and response variables, machine learning models have parallels to simpler sta-

tistical models like linear regression. However, machine learning models consist of hundreds, thousands, or millions of individual equations, while a linear regression model consists of only one equation. Because of their many equations, machine learning models are commonly considered "black boxes" since peering into a machine learning model can feel like looking into a pitch black box – you don't precisely know what's inside. Methods now exist to help us illuminate machine learning black boxes, and the field of "interpretable" or "explainable" machine learning has made great strides in support of better understanding the inner workings of these models (Samek et al., 2021). Still, machine learning models are substantially more challenging to interpret than most other model alternatives.

While challenges to interpretation are a clear pitfall of machine learning models, their complicated structures allow for them to pick up on relatively subtle or nuanced relationships between datasets, making them effective predictive tools. In many cases, machine learning models outperform process-based models (e.g., their predictions can have greater accuracy than the predictions from other models). The ability to create strong predictions is arguably the hallmark strength of machine learning. The previously described flexibility of being able to include many different diverse data types (e.g., images, sensor data, weather station observations) as predictors is also a key strength. But, because they do not necessarily account for underlying processes, machine learning models are at risk of spurious predictions, particularly if developed irresponsibly. In chapter 3, we include questions you can ask to evaluate whether a model was developed responsibly, and assess whether it is vulnerable to making spurious predictions.

2.3 Classes of machine learning models

In the previous example on image-based predictions of water velocity, we summarized the use of a *supervised* learning approach. Simply put, supervised learning is when there are true answers for the model, or there are measurements for the response variable (Sarker 2021). In the water velocity model example, water velocity measurements were used during model development to facilitate the learning of patterns in the images that could specifically be used to predict water velocity. Once a supervised learning model is developed, the predictions from the model can be compared with the measurements to assess how well the model predicts.

Unsupervised learning, on the other hand, performs tasks that do not have answers for the model. For example, instead of using the images of the stream to estimate water velocity, an unsupervised machine learning model could be used to group or cluster images with shared similarities. Unsupervised learning is often used as a computer-assisted way of exploring patterns in data (Sarker 2021). Sometimes, the groups or clusters uncovered by an unsupervised learning model can be assigned labels that are of use for other modeling or analysis

efforts.

2.4 An example of supervised and unsupervised learning

The National Land Cover Database or Dataset (NLCD) is created by the U.S. Geological Survey and its partners to map land cover across the contiguous U.S. The data are updated every few years, creating a historical record of land cover change across the country over time. In the 1970s and 1980s, land cover mapping was performed by manually delineating areas from aerial photographs. In 1992, the NLCD premiered a land cover data product that was created from Landsat satellite imagery. Landsat imagery has pixels that are 30 meters by 30 meters. To create the 1992 product, NLCD imagery was clustered into 100 groups using an unsupervised learning approach, and people evaluated the 100 groups and manually assigned them to land cover categories (e.g., forest, developed area). Later, to create the 2001 NLCD product, a supervised learning approach was used in which areas of known land cover were used to train a model to predict land cover categories based on Landsat imagery. Read more about the history of the NLCD program in Chapter 18 of *The Nature of Geographic Information* (DiBiase et al.).

2.5 Algorithms

Within the two broad classifications of machine learning (supervised and unsupervised), there is an overwhelming number of machine learning *algorithms*, or specific computer model frameworks for implementing machine learning. The sheer number of machine learning algorithms attests to its power and versatility. This primer does not catalog different machine learning algorithms. As a starting point, you can see some algorithm examples at the resources below. Which machine learning algorithm should I use by SAS blogs Machine Learning Cheat Sheet by Datacamp

2.6 A deeper dive

If you would like to learn more about machine learning, check out *An Introduction to Statistical Learning* by James et al. (2013). This book provides a wealth of information on machine learning models, and the authors assume the readers are mainly interested in applying, rather than deeply studying, machine learning models. The four premises offered by the authors, listed on pp 8-9, demonstrate the practical focus of the book: Many statistical learning methods are relevant

and useful in a wide range of academic and non-academic disciplines, beyond just the statistical sciences. Statistical learning should not be viewed as a series of black boxes. 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! We presume that the reader is interested in applying statistical learning methods to real-world problems. There are free PDF versions of the books available online, and examples are presented in both R and Python.

Chapter 3

Key considerations

Authors: Natalie G. Nelson, Shih-Ni Prim, Sheila Saia, Khara Grieger, Anders Huseth, Brian Reich

Here, we offer some key questions natural resource management practitioners can ask before using or developing a machine learning model to inform day-to-day decision-making. Importantly, these questions are based on our own opinions and experiences, and should not be taken as definitive criteria. The questions are provided to help you start thinking critically through the process of using machine learning model outputs.

Why was a machine learning model used instead of a process-based model or more easily interpretable statistical model? Because process-based or mechanistic models capture underlying physical, biological, and chemical processes driving agricultural and environmental phenomena, they are, in theory, less vulnerable to making spurious predictions as compared to machine learning models. However, process-based model development is often time- and resource-intensive, and some agricultural and environmental phenomena are not well predicted or explained by process-based models. Generally, the more a system is dominated by physical processes, the more predictable the system's behaviors are by process-based models (Haefner 2005); physical processes are often well represented by established mathematical formulations. On the other hand, for behaviors that are largely driven by system biology or ecology, process-based models can have poor performance because biological and ecological processes are more challenging to robustly and accurately represent with mathematical equations. For example, it is reasonable to assume that it will be easier to simulate changes in a lake's water levels than the amount of algae in the lake.

In cases when a physical or process-based model could have been developed instead of a machine learning model, it is reasonable to question whether the machine learning model is suitable to support decision-making. Similarly, if an easily interpretable model like a linear regression model can be used with

adequate performance, that is generally preferable, as model users will have greater understanding of how the model operates.

How much data are needed to develop a machine learning model? There is no minimum dataset size for developing a machine learning model, but machine learning models should generally only be developed from large datasets (e.g., thousands or millions of data points). Since machine learning models learn patterns in data, there has to be enough data from which patterns can be learned. Importantly, the data should also be of high accuracy to ensure the machine learning model is developed from quality information. Because of the dataset size requirements, in environmental and agricultural systems, we often see machine learning models being developed with data from sensors and imagery, as such data collection systems produce large volumes of data.

What data were used to train the model? This is probably the most important question to ask. Because machine learning models make predictions from patterns in data, the data provided to train the model dictate the types of predictions a machine learning model can make. If a training dataset is narrow (e.g., it includes a short time period, a time period with little environmental variation, a limited number of locations, etc.), its predictions will not generalize. If a machine learning model was developed with data from corn fields in the Midwest, will its predictions apply to corn fields in Georgia? In particular, ask yourself (or the model developers), what is missing from the training data? For example, if you're interested in predicting the effects of drought, how many drought periods are included in the data, and how severe were the droughts? Even if the training data are spatially extensive, think critically about whether the data may be biased or disproportionately unrepresentative in some way such that the data may not be generalizable.

For example, research has shown that surface water quality monitoring in the Southeast U.S. disproportionately occurs in more affluent areas (Oates et al., 2024), meaning that areas of greater social vulnerability have fewer monitoring stations. If a machine learning model were to be developed using all available water quality data in the Southeast for the purpose of predicting water quality in unmonitored areas, it might sound as though the training data are comprehensive for the region, but the model would ultimately be affected by the training data's underlying bias. If a model developer is unable to disclose the data sources (e.g., because of data ownership/privacy concerns), they should still be able to provide information on general data characteristics such that you can assess whether the training data apply to your system of interest.

How was the model tested or evaluated? Were any measures of uncertainty reported? To understand how well a machine learning model performs, it is important to understand the model's error, or the difference between the model's predictions and true measurements. Error can be reported in many ways. One common metric is the Root Mean Square Error (RMSE), which is essentially the average model error. The RMSE is reported in the same units as the target variable, making it useful to interpret. Metrics like the coefficient

of determination, or R2, are also commonly reported. Learn more about commonly used error metrics here. Understanding a model's error is key to assessing its accuracy and reliability. In addition to error, modelers will ideally report measures of uncertainty. Uncertainty describes a range of plausible model outcomes. For example, let's say you are running a machine learning model that can predict daily plant water demand and one of the predictors is rainfall. If you expect there could be 1-1.5" of rain today, you could run the plant water demand model with there being 1" of rain today, or 1.5" of rain. Both are plausible. When you run the model for the two scenarios, the difference in the predicted plant water demand would be described as the uncertainty in the model output. Consideration of uncertainty is a hallmark sign of responsible modeling.

Are the model developers transparent about limitations and model performance? Be wary of salesmanship. Responsible modelers will clearly articulate limitations of the training data, and aspects of poor model performance. A model developer should be able to tell you if the model is effective or not at predicting different types of outcomes.

How did the modelers choose predictors? Do they seem to understand some of the underlying science or processes of the agricultural or environmental system they are making predictions for? Ideally, when predicting agricultural or environmental system behaviors in a machine learning model, there should be sound scientific reasoning regarding the selection of predictors. Judicious selection of predictors can also help to avoid spurious predictions.

What do the modelers know about predictor importance? Once a machine learning model is developed, there are methods available for identifying the predictors that carry the most weight, or are the most "important", in the model. It can be helpful to understand which predictors are the most important, as you can then use your own understanding of environmental and agricultural system dynamics to corroborate whether the variables of most importance make sense. If an obscure variable is the most important in the model, it's reasonable to question why, and whether the importance of an obscure predictor could negatively affect model performance.

How will the outputs of the model be used? Machine learning models should not be used to replace human decision-making in natural resources management, but can instead be used as a supportive tool. Machine learning models can synthesize a lot of information and identify patterns that are not readily interpretable for the average person, making them great decision-support tools. However, keep in mind that training data are almost always biased in some way, and those biases will propagate to the predictions.

How will the model be run, or how will the outputs be shared? Developing a machine learning model and creating a user-friendly system for applying and/or accessing outputs from a machine learning model are entirely different

tasks. In some cases, a model can be developed and then shared via computer code, but end users may struggle to run the model using provided computer code, depending on their familiarity with programming. If you are working with a partner (e.g., university, consulting group) to produce a model, ensure there is a plan for transitioning the model such that you can use it with ease. Our team has created a resource for those interested in creating apps for disseminating models, Ten Simple Rules for Researchers Who Want to Develop Web Apps, which may provide helpful tips if your team is interested in exploring the use of web apps for model and model output dissemination.

Chapter 4

Case studies

Authors: Natalie G. Nelson, Shih-Ni Prim, Sheila Saia, Khara Grieger, Anders Huseth, Brian Reich

To illustrate the use of machine learning across diverse natural resource management applications, three case studies are presented here, all of which use the same machine learning modeling framework. These three case studies include models for predicting (1) daily pluvial flood dynamics in agricultural landscapes, (2) fecal coliform exceedances in shellfish growing waters, and (3) sweetpotato yields at the county-scale. All three models use Random Forest, a specific type of machine learning algorithm. Random Forest creates many (e.g., hundreds or thousands) of regression or classification trees to predict categories or continuous values of the response variable. Random Forest is considered a particularly effective algorithm for working with environmental and agricultural data, as it can effectively capture nonlinear and complex interactions between variables. All three case studies offer interpretations of the most important predictor variables in the model, and explanations as to why those predictors may be the most important.

4.1 1: Daily pluvial flood prediction

Machine learning approach for modeling daily pluvial flood dynamics in agricultural landscapes (Fidan et al., 2023)

- Response variable: Binary categories of "flooded" or "not flooded"
- Predictor variables: Soil characteristics (flood frequency, drainage class), elevation, height above nearest drainage, slope, topographic wetness index, distance to nearest stream, distance to nearest road, population, multi-day precipitation

In this study, the model developers focused on an agricultural area that experienced pluvial, or rain-driven, flooding following a major hurricane. The area was treated as a grid, where the grid was divided into square pixels of equal area. The model was developed to estimate the likelihood of a pixel being flooded or not flooded based on landscape characteristics (e.g., soils, land cover type), and recent rainfall patterns.

Read the full study here.

4.2 2: Fecal coliform forecasting

Short-term forecasting of fecal coliforms in shellfish growing waters (Chazal et al., 2024)

- Response variable: Fecal coliform (bacteria) concentrations
- Predictor variables: Rainfall, river stage, wind speed and direction, lengths of artificial and natural channelization, land use and land cover, soil drainage class, air and water temperatures, month (i.e., time of year)

In this study, the model developers used long-term monitoring data from the Florida Department of Agriculture and Consumer Services shellfish sanitation program. The monitoring data were collected at specific sampling locations across the coast of Florida where shellfish are grown for harvest. The model was developed to predict the likelihood of a fecal coliform (i.e., bacteria) concentration exceedance at each station where routine monitoring data are collected. Predictors in the model included a range of variables that could explain the source and transport of the bacteria, as well as factors that could affect the survivability of the bacteria in the water (e.g., temperature).

Read the full study here.

4.3 3: In-season sweetpotato yield forecasting

In-season Sweetpotato Yield Forecasting using Multitemporal Remote Sensing Environmental Observations and Machine Learning (Carbajal Carrasco et al., 2024)

- Response variable: Sweetpotato yields at the county scale, reported by the USDA National Agricultural Statistics Service
- Predictor variables: Elevation, slope, aspect, soil characteristics (% clay, % sand, pH, cation exchange capacity, bulk density, nitrogen, organic carbon), rainfall, maximum temperature, minimum temperature, and the Normalized Difference Vegetation Index calculated from satellite imagery (Landsat and Sentinel-2)

In this study, a model was developed to predict end-of-season sweetpotato yields at the county scale using early season predictors, such that the model can be used to forecast harvested yields early in the growing season. The model uses a range of predictors that capture climatic and physical properties that influence sweetpotato yields over large spatial scales, but does not account for cultural or management practices.

Read the full study here. Note that this study is currently undergoing peer review, and its findings should be considered preliminary.

Chapter 5

Workflow demonstration

Authors: Shih-Ni Prim

In this section, we use a dataset to exemplify a typical workflow for constructing machine learning models. We skip exploratory data analysis, or the process of exploring your data to understand its structure and components, but such exploration should always occur prior to any modeling. For more on exploratory data analysis, refer to Chapter 4 Exploratory Data Analysis in *The Art of Data Science* by Roger D. Peng and Elizabeth Matsui.

The dataset analyzed here contains wine quality and traits, and our goal is to predict the quality of wine using traits. The dataset can be found here.

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

5.1 Prepare the data

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

```
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.
```

```
# 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)
```

Here, we'll develop a Random Forest model, which is the same machine learning algorithm used to develop the models presented in Chapter 4. To demonstrate how to construct a Random Forest model 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.

```
# 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
```

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

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

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

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

Next, we randomly separate the dataset into a training set (80% of the rows)

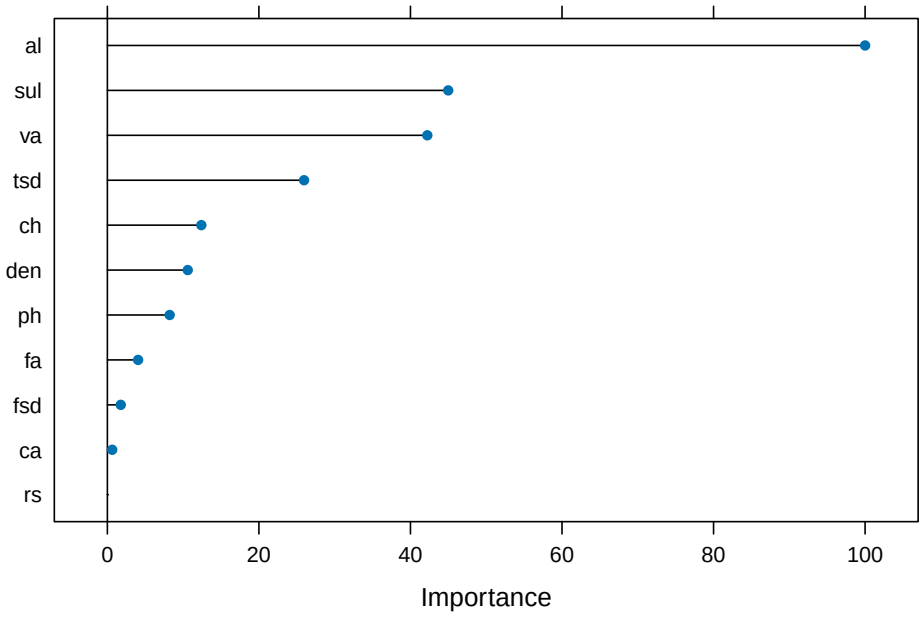
and a testing set (20% of the rows). A seed is set so that the result can be reproduced.

```
# separate data into a training set and a test set
set.seed(2)
train <- sample(nrow(winez), nrow(winez)*0.8)
winez_train <- winez[train,]
winez_test <- winez[-train,]
```

5.2 Classification model

Now, we will construct a Random Forest model with a two-level response. After the model is run, we calculate the predictions using the `predict` function. A plot with the most important predictors in the model is provided. We can see that `alcohol`, `sulphates`, and `volatile acidity` are the 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))
```



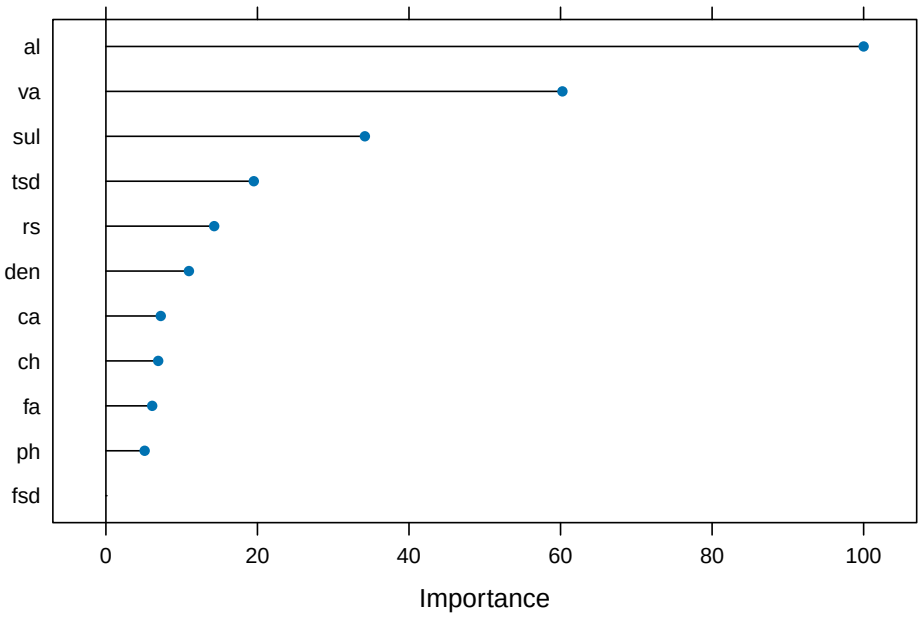

```
confusionMatrix(rf2_pred, winez_test$qual2)
```

```
## 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)
```

5.3 Continuous model

Next, we construct a Random Forest 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 the mean squared error (MSE), find important variables, and show how the error decreases with the number of trees constructed. The three most important variables are again `alcohol`, `sulphates`, and `volatile acidity`.

```
# 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)
```

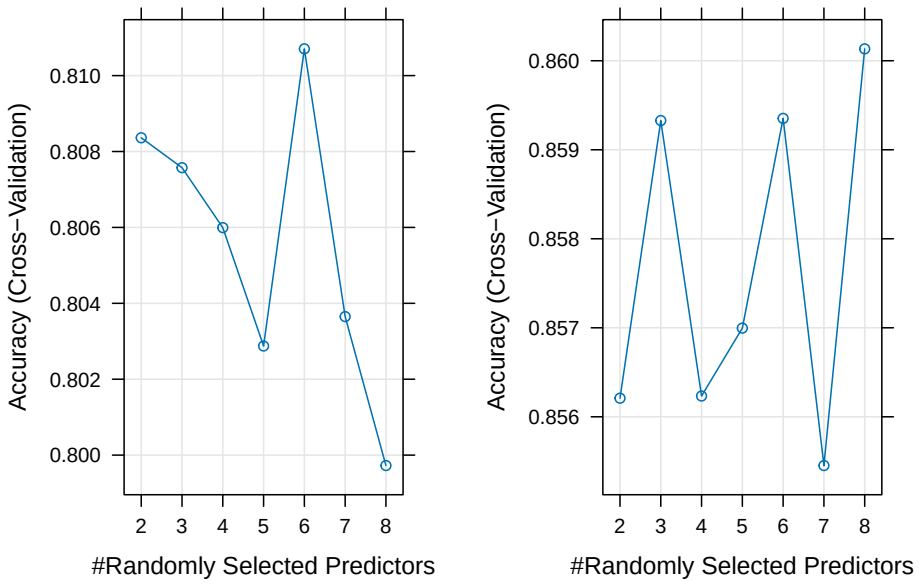
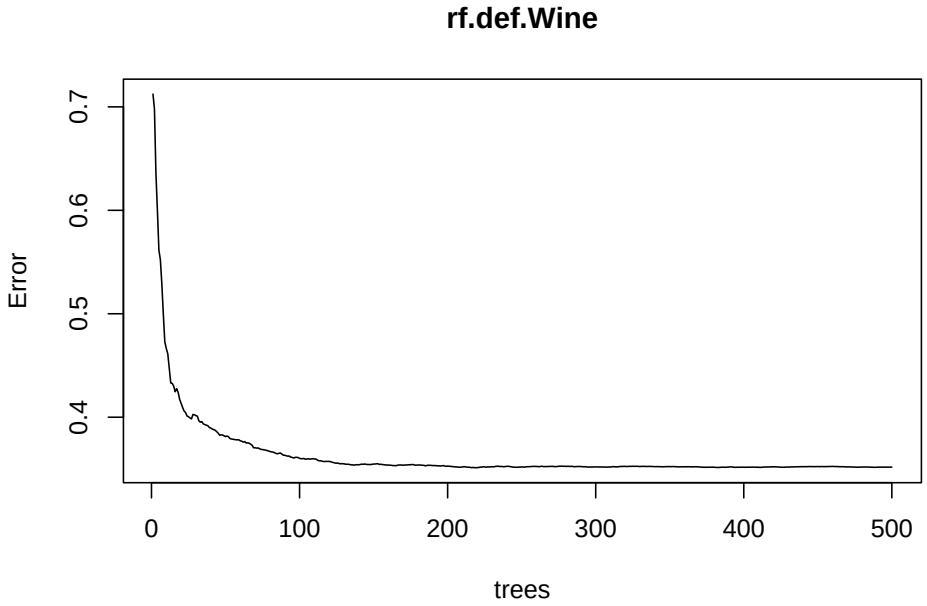


Figure 5.1: Accuracy Rates vs Number of Predictors Used for 2-level Variable (left) and 3-level Variable (right)

```
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
```

```
plot(rf.def.Wine)
```



We try several different values for the number of variables 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 + s
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)
```

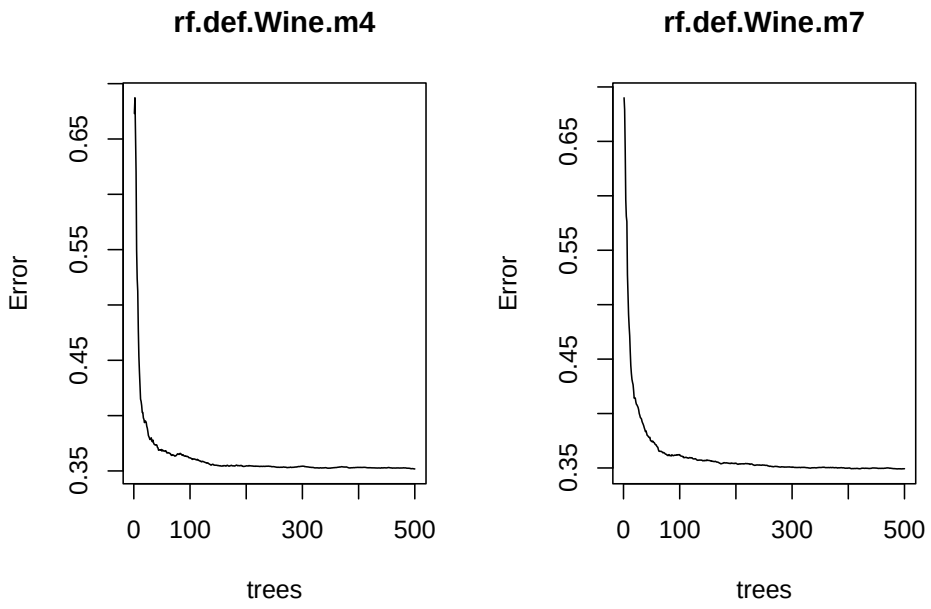
```
##      Overall
## fa  22.78226
## va  39.40868
## ca  22.22551
## rs  16.27685
## ch  25.12775
## fsd 21.58168
## tsd 30.36789
## den 27.20655
## ph  20.22223
## sul 47.59070
## al  60.54909
```

```
# mtry = 7
rf.def.Wine.m7 <- randomForest(qual ~ fa + va + ca + rs + ch + fsd + tsd + den + ph + s
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
```

```
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.

```
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