A ‘Day 1’ Intro to Machine Learning

Benjamin Manning, PhD

Feb 7, 2019 · 10 min read

In my last post, A ‘Day 0’ Intro to Machine Learning, I introduced the very beginner to some high-level concepts in Machine Learning (ML) and did my best to correlate some of the methodology behind the scenes of Supervised ML to a common skill many of us likely learned as a kid: learning to ride a bike.

Now that the foundation has been laid, lets start building the remainder of the structure and dive a little deeper into the processes within a typical workflow.

As we go through these processes in more detail, I will occasionally re-introduce aspects of our previously learned skill (bike riding) to continue making the comparisons with the high-level concepts we’ve already covered.

Last point of note: this blog is mainly for supervised ML, but I’ll try to point out common areas the processes have between the other ML methods as we make the journey.

A. Data

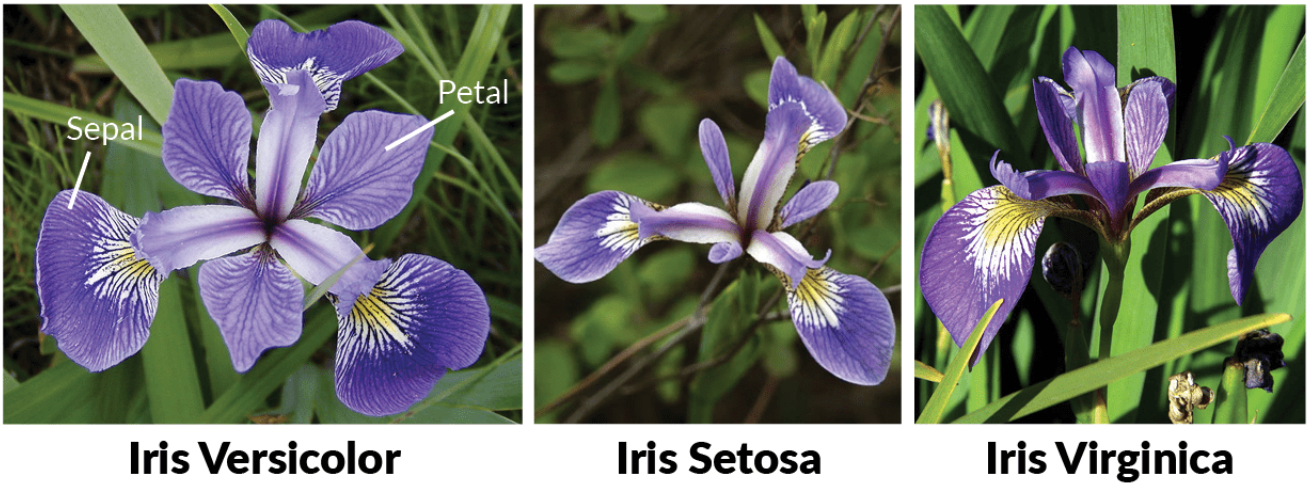
Before we begin the next step in our journey, we’re really at the point where we need to start working with some data to fully understand what we’re doing in each part of the process. Many introduction-type posts use the Iris Dataset (Fisher, 1936) and that is what we’ll be using here, so let’s start by obtaining a better understanding of the data from a domain view.

Note: For beginners, this is one of the most overlooked steps in the entire ML process and even more so for Data Science beginners. Always remember, when available, ensure you take the time to educate yourselves on the domain of the data (ie. Where did it comes from? and What does it describe?). Knowing this type of information ahead of the project will help to better define how you go about solving it.

According to the UCI Machine Learning Library:

> This is perhaps the best known database to be found in the pattern recognition literature. Fisher’s paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the

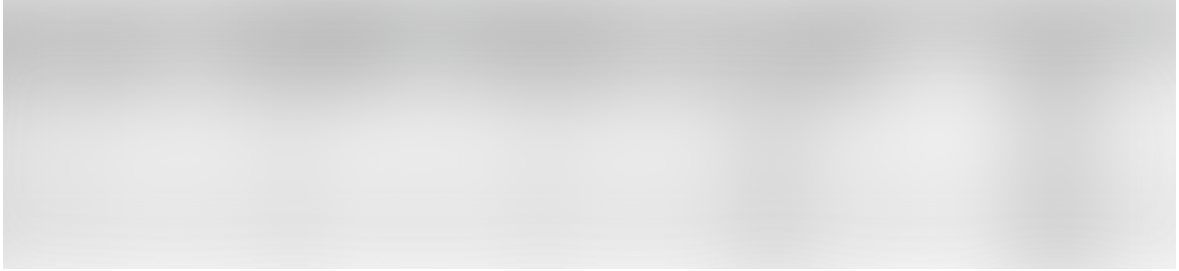
> other 2; the latter are NOT linearly separable from each other. *Predicted attribute: class of iris plant.*



Iris Flowers in the Iris Data

So what does all of this mean? Basically, this data quantifies the physical characteristics of three different type of Iris flowers. As you can see in the table below, the ***length*** and ***width*** of the *Sepal and Petals* have been recorded. These will represent portions or elements of the experiences we described in our earlier discussion — remember learning to ride your bike? Each time you likely tried different things in order to successfully ride your bike or ***get to the outcome***. For every iteration, you likely made small adjustments in the same areas and gauged the impact of the changes each time you tried something different. This trial and error approach is a common way we learn new things, but we have to capture or quantify these small changes in ***order for ML to Learn*** which corresponding combination equated to a specific outcome.

This problem describes the basic structure of most supervised ML problems: we have the many different measured elements, commonly called **features or independent variables**, of each observation (row in the data) and a single outcome, commonly called the **dependent variable**, related to each set of measurements as shown below.



Iris Dataset Variables

Note for beginners: In this problem, we know what the features are, but in more complex ML problems we do not have this luxury. In fact, this is the most important part of the entire ML process — feature selection and feature engineering is a complex and often complicated science that deserves much more focus than what we can provide here. My main point here is the heavy lifting has been done for us because the data is well known — this is very often not the case in ML or Data Science (See EDA in the previous post).

B. Preprocessing

It is commonly said that 80% of the work in the ML process is done in this area. The bottom line is: if we do not have clean data to work with the entire ML process will be faulty. One benefit of the Iris Dataset is it has already been cleaned, but here are some examples of what else could be done in this phase.

B.1: Missing Data

In the table below, you can see there are some cells with missing data. This is commonly referred to as NAs in ML and/or Data Science and, while some ML algorithms will work around NAs, we still bear the burden of deciding what needs to be done with data that has no value. The process of handling missing values in data is called **imputation (**Van Buuren, 2018).

Missing Data

In Statistics, there are numerous ways to impute missing values, but in ML or Data Science the most commonly accepted method when working with numeric data is to replace missing values with the mean value of the numeric feature. This ‘good enough’ approach is helpful, but should be used with caution and only when there is not sufficient domain knowledge to make a substantive replacement.

B.2 Data Types

Understanding that different data falls into different types is fundamental to ML and nominal and numeric data were both lightly introduced topically in the prior post. Taking for granted that your tool of choice will assign the correct data type needed to each variable is a common mistake beginners often make. For example: when the Iris Dataset is imported into R verbatim, this is what is assigned to each variable:

Species as a Character Datatype

Notice that ‘species’ was imported as a ‘chr’ datatype? While this is technically correct, it does us no good from an ML aspect since ‘chr’ is not anything we can use in supervised ML methods like classification that use a nominal dependent variable. Notice, after converting the variable to a factor (nominal data in R) the specific classes (each type of Iris flower) are now found by R as shown below as ‘levels’ or categories or classes in ‘species’:

Species as a Factor or Nominal Datatype

C. EDA

In our previous post, this section was described as follows: *this process of learning as much as possible about the data is called Exploratory Data Analysis or EDA for short. We use*

all sorts of visualizations and tools that can help us accomplish this, but, for now, just know why we do this — to learn what we do not know.

We have already educated ourselves about the domain space of the data and any high level relevant information that was available about the data. In EDA, we dive into the Statistics of the data so we can begin to unfold the quantitative information better and begin to design a proper way to define and eventually solve the problem.

There are two main ways to dive into this portion of the process: functional and visual. Functional methods are provided by almost every tool or language used for ML or Data Science and provide a high level understanding of the main Descriptive Statistics in the data. For example: R uses summary() to provide the high level details as shown below:

Summary() of Iris

The pandas Python library uses a similar function called describe() to accomplish a similar task as follows (notice the absence of the Nominal variable):

Describe() of Iris

Visual methods can also be very useful for obtaining a better understanding of the data and we can easily use the more typical tools available in almost all ML packages to plot different elements of the data. For example: the Iris Dataset description describes the following “***One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.”*** If we wanted to ‘see’ how this might impact the other features, we can create a scatter plot that shows the interaction like the one below.

Notice the *Setosa* flower is completely separated and the remaining two flowers’ features not only overlap, which conveys some type of similarity ***(not causation, be careful not to assume this)***, but are also somewhat positively correlated in their dimensions.

So why is all of this important? Lets go back to the bike example to close the loop on this.

Every time you tried to ride your bike while learning to do so, you learned something new that you likely applied to the very next iteration. Maybe it was something new to try or maybe it was something you needed to stop doing — ***the point here is you were acquiring New Knowledge about the problem every time you tried — and this is how***

you eventually formulated a method for solving the problem even if it was from small victories one at a time.

This is what we’re doing with EDA — we’re trying to gather all of the necessary details about the data so we can: **A.** make an informed decision about creating the feature space (we already know this in this problem) and **B.** we can make a decision about the type of problem this will be and how we will solve it (which methods will likely work best).

D. Modeling

Now that we have a better understanding of our data from both the domain side and the Statistical side, we can begin to try a few models. As beginners, it is best to follow an approach I like to call ‘push and turn’ . If you are like me, when you get something new or are trying something new you ‘push and turn’ every button or try every iteration to see how the outcomes differ each time (i.e. What happens when I ‘push or turn’ this?)

While this might seem like overkill in ML, it’s a good method to follow when you are learning so you experience how different algorithms treat the same type of data.

Below, I have tried both an MLP and KNN on the data and found KNN to be more sensitive to the data than the basic Neural Network was with a higher accuracy of .980. ***(There is MUCH more to this than I let on here and we’ll dive deeper in the next post).***

Model Trained with a MLP

Model Trained with KNN

After a quick check comparing the predictions made from the trained KNN model with the ground truth in the testing set, I can easily check for overfit and ensure the model will generalize as needed.

Final note for beginners: This post is directed at *YOU* and I hope you learn from it, but keep in mind this is a very basic exercise meant for learning — after all, it is only your *second day*! In the next blog post, Ill dive into the heart of ML and discuss model training, testing and deployment in much more detail.

Some of the portions of this blog post can also been seen in this video:

References

Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0–471–22361–1. See page 218.

Fisher, R. (1936). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Van Buuren, S. (2018). Flexible imputation of missing data. Chapman and Hall/CRC.

. . .

*Feel free to share on other channels and be sure and keep up with all new content from Hashmap on Medium — you can follow us **here**.*

*Dr. Benjamin Manning is the Lead Data Scientist at **Hashmap** specializing in growth across all industries and partners served by Hashmap. He has been a Machine Learning and Data Analytics consultant for seventeen years and specializes in engineering disciplines such as Solar Energy, Oil and Gas and Iot/IIot Systems.*

Ben is also faculty member of the College of Engineering at the University of Georgia where he teaches courses in Data Science and Computer Systems Engineering while also serving as the faculty advisor for the university’s student informatics club: Data Dawgs. Ben is the Senior Data Science Mentor for Experiential Teaching Online and teaches Data Science online for the University of Texas-Austin and Rutgers University.

You can connect with him on LinkedIn or email him at benjamin.manning@hashmapinc.com

Machine Learning Data Science Education Data Analytics Storytelling