


Ironhack Student Portal

 preview.my.ironhack.com/lms/courses/course-v1:IRONHACK+DAPT+201910_MAD/units/4522c92ec252408981e64e0428e3fe21

Machine Learning Workflow

Learning Goals

- Learn the different ways of obtaining data for supervised learning
- Understand the processes involved in learning from data (training, testing, and evaluation)
- Know why data cleaning and preparation are necessary

So far, we have gone over some different types of machine learning, and a general overview of what is machine learning. Now we are going to go more in-depth about exactly how machine learning works on a practical level.

Workflow

Here, we are going to define a simple outline of the machine learning workflow. While different subsections might be added or removed depending on the goal or complexity of the project, the overall structure remains the same.

Question Formulation → Obtain Data → Explore & Clean Data + Feature Engineering → Select & Train Model → Obtain & Present Results

Question Formulation

Perhaps the most important part of any machine learning project is formulating a question. The question we ask will help to know how to approach our dataset, and what analyses to perform on it. Without a well-defined question, it is hard to know where to start and easy to get lost in all the possibilities of what you could do with a particular dataset. Optimally, the question tells us about both the **output**, the result that we want to obtain, as well as the **input**, the data that we have to help us answer the question. Let's take a look at some examples.

Question: *Can we predict what type of animal is in this picture?*

Input: Pictures of animals **Output:** Classification of animal into its type

Question: *Can we predict stock prices based on the trends we have seen this year?*

Input: This year's stock prices **Output:** Prediction of future stock prices

In the examples above, we can see that including the input and output in the question helped us to very clearly define our goal: we know what we want our result to look like, and have first idea of the data we are going to use to achieve it.

Obtaining Data

Data is a basic necessity for machine learning. Whether we have data with a labelled outcome, or data that does not have a labelled outcome, or we have millions of observations or hundreds, data is an essential requirement to implement any kind of machine learning. We are going to focus first on the data required for Supervised Learning, as there requirements of the dataset are higher, given the need for labels.

Let's imagine that we would like to solve a classification or regression problem (based on its attributes, is this animal a dog or a cat? Based on its characteristics, what is the likely price of this house?). We will need to have a dataset that contains observations with both the attributes and outcome labels (dog/cat or price of the house). Where can we find these data? Generally, there are three sources of labelled data, explained below.

Historical data: historical data contains information on both the features of the observations and the outcome, and has been collected over some period of time. Weather data is a good example of data that has been collected for a long period of time. For example, if we wanted to be able to predict rainfall in a certain area, a weather data would provide us with a good dataset with which to start working. It is not necessary that historical data (or any dataset) contain only features relevant to the question that you would like to answer: some features of a weather dataset might be highly related to rainfall (humidity, pressure, temperature, sunshine) and others that are not related.

Auto-labelled data: Auto-labeling is a way to build a labeled dataset by using a tool or process. Let's imagine for instance, we would like to build a dataset on which to train a model that can distinguish between pictures of dogs and cats. To save ourselves the work of manually tagging pictures of dogs and cats, we could build a data set by using the image search results from googling 'dog' and 'cat'. This is one way we could end up with the dataset we need, by using a tool that provides the labels for us. Similarly, we could scrape Instagram and classify all those photos with the hashtag #doggo as dogs and all those with the hashtag #instacat as cats.

Manual-labelled data: The final way of obtaining labeled data that we are going to explore is through manual labelling. This method requires a human analyst to label the data we have to provide the outcome class. It is usually the most time and labor intensive, and for that reason it is hard to accomplish for very large datasets. For example, imagine that our task is to detect extremist speech in a language that we do not speak. While we could use translation, a more accurate option to build our training dataset would be to employ analysts who speak the language to determine if the data in our sample contains extremist speech. One technique that makes this type of labelling easier is crowdsourcing. For instance, Wikipedia provides a dataset that is used to detect hate speech. This dataset was compiled through aggregating data from all of the comments that different users had reported as containing hate speech over several years.

We can get labelled data in different ways: by already having labels in the historical data that we have, auto-labelling our data, or manually labelling the data that we have.

BEWARE! In addition to understanding the data we have and if we need to label it, it is important to understand the form of your data and what it means. Do you know the **format** the data is in? Do you need to make a plan to **merge or concatenate** many separate datasets? Are you able to **load the data** correctly? Can you **query** it to obtain a subset that is relevant to your question? Is there a **codebook** you should be aware of to be able to understand the variables in your dataset? These are all important questions to ask to make sure that you have appropriate and usable data.

Data Cleaning & Exploration

So, as soon as we have labelled data, we can through it into our algorithm and it will start learning, right? Wrong! We have to complete several steps to prepare our data to be learned from first. We are going to discuss some of the steps you might want to implement depending on the data you have. Don't worry about understanding exactly when to use each process quite yet-- we will discuss them in greater detail in later lessons.

Preparing the data to be processed by algorithm is called **data cleaning and feature engineering**. This is the same process that we completed before doing analyses, but for machine learning it often includes a few more steps.

One technique we might employ when cleaning our data is **unit conversion**. Imagine that we run two linear regressions, and want to interpret their outputs. In the first, we have measured distance in kilometers, and find that the distance from the city center has a huge effect on the price of a house (for every 1 km increase, the price of the house decreases by \$40,000). In our second regression, we measure distance in centimeters. Though this is a bit unrealistic, you might imagine that 'distance' now has a much smaller effect size: for every 1cm increase in distance from the center, the home price only decreases \$40. While we know that these two examples mean the same thing ($\$40,000/1,000 \text{ cm} = \$40/1 \text{ cm}$), the difference might lead us to believe incorrect conclusions about the effect size of distance -- especially if there are other distance variables. Moreover, because many machine learning algorithms' interpretations are not as straight forward as that of linear regression, if we do not measure variables using the same units (where possible and appropriate), those x variables with smaller units will exert a larger influence on the outcome, just like we saw with kilometers.

Another approach to cleaning our data might be binning. **Binning is the process of grouping similar elements in one feature together to make a new feature based on the same information, but with less granularity.** Binning can be done on both categorical variables (poodle, pug, siamese, sphynx, beagle, munchkin... can be grouped in to bins of "cats" and "dogs") and numerical variables (11, 12, 8, 15, 23... can be grouped into bins of 0-10, 11-20, 20-30, etc). Binning reduces the granularity of the data,

which helps the algorithm to avoid overfitting (it will no longer take into account the difference between pugs and poodles, or between 11 and 12 in the above examples). Of course, this comes with a cost to performance due to the loss of granularity.

Another technique we have to preprocess the data is called **feature scaling**. In datasets, features often have very different ranges, magnitudes, and scales. For example, following the housing example, we could include 'distance to the city center' in kilometers, but 'garage size' in terms of the number of cars that fit. This means that distance to the city center would have a range between 0 and several hundred, whereas garage size will have a range from 0 to something around 5. For many machine learning algorithms that are based on measuring distance, we input these features, 'distance' will have a much larger influence on the outcome, just because of its larger magnitude. Unfortunately in this case, there is no way to measure garage size in kilometers with the information we have. This is where feature scaling comes in: feature scaling puts all of variables on the same level of magnitude. We will discuss exactly how this is achieved in future lessons.

Often in working with data, we find that our dataset contains **missing values**. These values could be due to mistakes in the data, or because the value of that particular X variable is unavailable for that observation. It is important to know, or be able to make a good guess, if these missing values are random or if there is a reason why they are missing.

If we know or believe that the missing values are randomly occurring in the dataset, we have several options: we could drop these rows from the dataset or we could fill them with the median or mean of the variable value. These methods are best used when there are not very many missing values, so as not to greatly reduce the size or variability in the dataset.

If we do not think that the values are missing randomly, we need to be careful about how we treat these missing values. Is there something in the data codebook (explanation of the data) or something we know about the data that could help us fill these values? For instance, maybe we have many missing values for the variable 'pool square footage' in our housing dataset. Maybe the houses that do not have this value are those houses that do not have pools? We should check this theory with the codebook or by looking to see if there are houses for which the pool square footage is 0 before filling in these data! In other cases, we might want to group our data by a different variable and then fill the missing values using the mean or median of the variable with the missing values based on this group. It is important to investigate well where there are missing values in your data, to make sure that the method you choose is not introducing bias into your dataset!

So far in statistics, we have only been working with numbers. But what if our data is not in the form of numbers? In many datasets we encounter variables that are categories (house type: apartment, single-family, duplex) or question answers that are in Yes/No/Maybe form. In order to handle data of this form, called **categorical data**, we have to find a way to convert it to numbers. We do this by building a new column for each of the categories of the variable. For each observation, we put a 1 where the

category of the categorical variable corresponds to the value represented by the column we made. This process of converting categorical data to numbers is called **one-hot encoding**. Below is a visual of how this process works:

color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

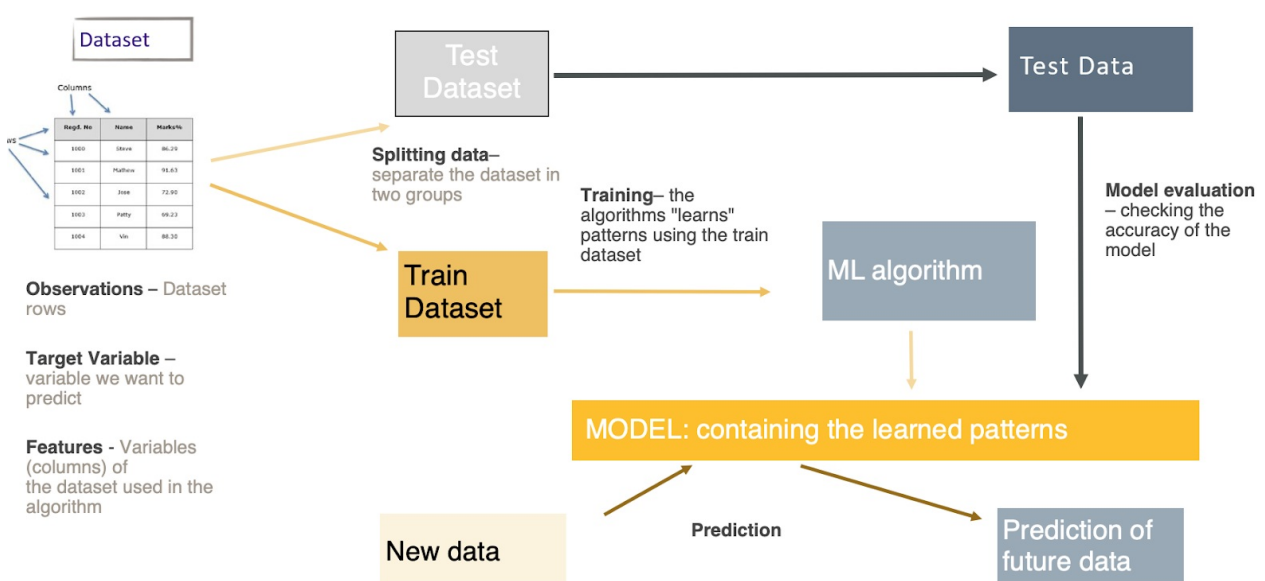
<https://www.machinelearningplus.com/machine-learning/caret-package/attachment/one-hot-encoding/>

For right now, we are going to stop with just data cleaning. We will look into data exploration in future lectures.

Select & Train Model

Selection: An important part of building *anything* is **choosing the right tool for the job**. In machine learning, this means selecting the correct model to use. Each algorithm has its own parameters and particularities, which mean that some work better than others in particular contexts. Once we have discussed different models in the upcoming lectures, this will become more clear.

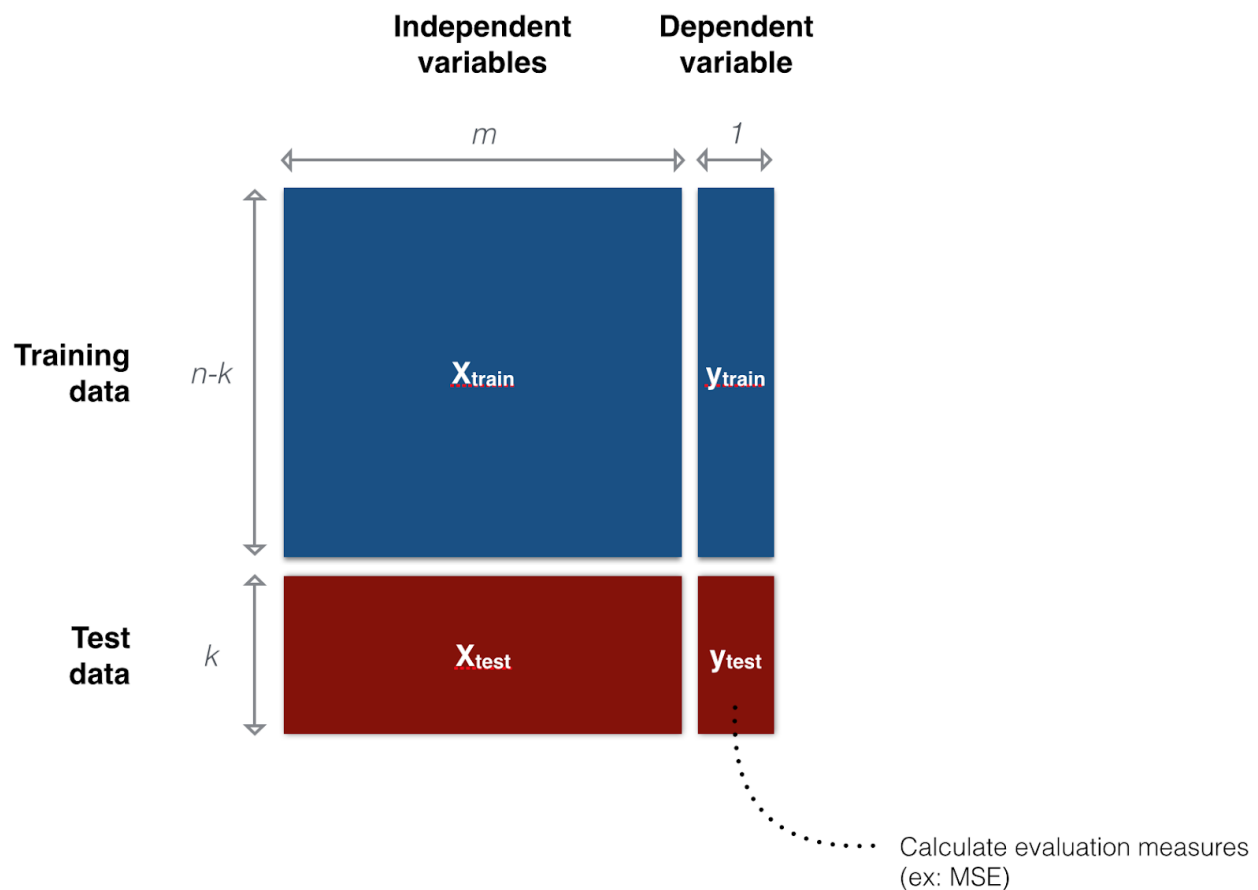
Let's look at the following diagram to understand an overview of model training:



Training: We have already discussed that when applied to data, algorithms learn those patterns in the data that determine the outcome in which we are interested. But how does it do this? During training, the algorithm is exposed to the different features that make up the training dataset as well as the associated outcome value for each set (row) of feature values. The algorithm learns by adjusting its parameters to best fit the data to which it is exposed. You can think of each row of the dataset containing some information about the features determine the outcome.

Showing more rows of data to the algorithm will make it easier for it to pick up on the signal (opposed to the noise) in the data. Training data is only 60-80% of the total data available. Why is that the case? This is because we reserve some percentage of our data in order to test how well the algorithm is performing. This is called the **test set**. If we let the algorithm see all of our data in the training set, we would not have any 'new' data for which we know the ground truth labels to show it in order to determine how well the algorithm has picked up on the signal in the data.

Like we stated above, we want to use the test set to understand how well the algorithm has 'understood' the signal in the training data. Therefore, the Y values from the test data act like the 'correct answers' for the test set-- if we want to know how well the algorithm performs, we have to remove them from the the test. In this case, we are asking the algorithm to guess, using ONLY the features (X variables) included in the dataset to predict what the outcome (value of Y) for each row would be. When we obtain these values, or *predictions* for the test set, we can compare these to the true values of Y that we have removed from the test set. Comparing these two values is how we can **evaluate** the algorithm, or determine how well it makes predictions.



<https://www.lucypark.kr/courses/2015-dm/multiple-linear-regression.html>

Supervised learning requires us to break our original dataset into several chunks: the training set, from which our algorithm will learn, the test set, from which we remove the outcome variable and on which we ask our algorithm to predict, and the outcome variable from the test set, which we use to evaluate those predictions.

Obtain & Present Results

Just like in data analysis, it is important we know how to present the results we find to our stakeholders. We should always keep in mind those principals of explaining our process: what question we were trying to answer, where our data came from, the processing we did to our data, the models we used, and why our results are important. In Machine Learning, we also want to **provide a measure of the performance of our algorithm**, which we will address in a future lecture on evaluation.

Lesson Recap:

- We have learned about three mechanisms to attain labelled data: auto-labelling, manual-labelling, and already having the data labels.
- Once we have that data, we discussed how algorithms work, and how the structure of machine learning helps us to be able to evaluate how well our algorithm is performing.

- Data cleaning & processing is a crucially important step to achieving accurate results. We have learned about converting units, feature scaling, imputing missing values, and dealing with categorical variables.