Key Points to Remember

We'll cover the following

Machine learning algorithms come with the promise of being able to figure out how to perform important tasks by learning from data, i.e., generalizing from examples without being explicitly told what to do. This means that the higher the amount of data, the more ambitious problems can be tackled by these algorithms. However, developing successful machine learning applications requires quite some “black art” that is hard to find.

Let’s go through some of the **lessons learned by machine learning researchers and practitioners** (put together in a great research paper by Professor Pedro Domingos), so that we can **avoid some of the major pitfalls**.

1. It's Generalization That Counts

The fundamental goal of machine learning is to generalize beyond the examples in the training set. No matter how much data we have, it is very unlikely that we will see those exact examples again at test time. Doing well on the training set is easy. The most common mistake among beginners is to test on the training data and have the illusion of success. If the chosen classifier is then tested on new data, it is often no better than random guessing. So, set some of the data aside from the beginning, and only use it to test your chosen classifier at the very end, followed by learning your final classifier on the whole data.

Of course, holding out data reduces the amount available for training. This can be mitigated by doing **cross-validation**: randomly dividing your training data into subsets, holding out each one while training on the rest, testing each learned classifier on the unseen examples, and averaging the results to see how well the particular parameter setting does.

2. Data Alone Is Not Enough

When **generalization is the goal**, we bump into another major consequence: data alone is not enough, no matter how much of it you have. Very general assumptions, like similar examples having similar classes, are a large reason why machine learning has been so successful.

Domain knowledge and an understanding of our data are crucial in making the right assumptions. The need for knowledge in learning should not be surprising. Machine learning is not magic; it can't get something from nothing. It does is get more from less though. Programming, like all engineering, is a lot of work: we have to build everything from scratch. Learning is more like farming, which lets nature do most of the work. Farmers combine seeds with nutrients to grow crops. Learners combine knowledge with data to grow programs.

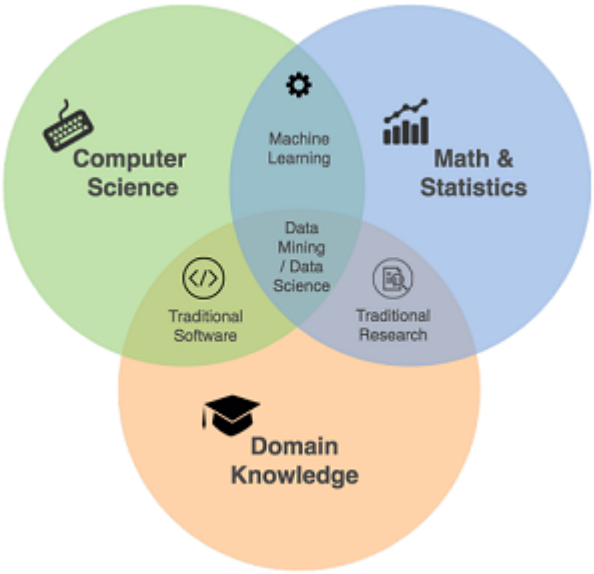


Image Credits: Machine Learning for Biomedical Applications: From Crowdsourcing to Deep Learning; http://mediatum.ub.tum.de/doc/1368117/47614.pdf

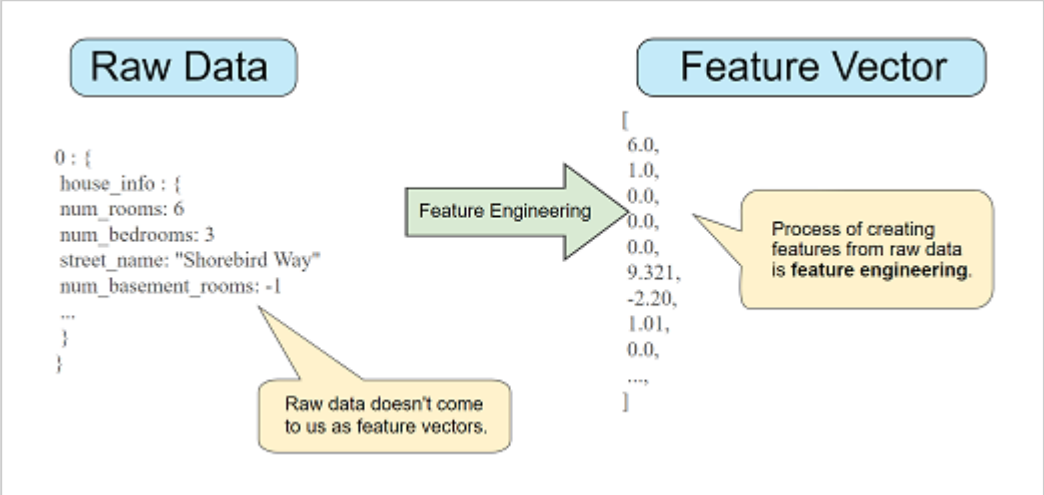
3. Feature Engineering Is the Key

At the end of the day, **some machine learning projects succeed and some fail. What makes the difference?** Easily the most important factor is the features used. If we have many independent features that correlate well with the class, learning is easy. On the other hand, if the class is based on a recipe that requires handling the ingredients in a complex way before they can be used, things become harder. **Feature engineering is basically about creating new input features from your existing ones**.

Very often the raw data does not even come in a form ready for learning. But we can construct features from it that can be used for learning. In fact, **this is typically where most of the effort in a machine learning project goes**. It is often also one of the most interesting parts, where intuition, creativity and "black art" are as important as the technical stuff.

First-timers are often surprised by how little time in a machine learning project is spent actually doing machine learning. But it makes sense if you consider how time-consuming it is to gather data, integrate it, clean it, and pre-process it, and how much trial and error can go into feature design. Also, machine learning is not a one-shot process of building a dataset and running a learner, but rather an **iterative process** of running the learner, analyzing the results, modifying the data and/or the learner, and repeating. Learning is often the quickest part of this, but that's because we've already mastered it

pretty well! **Feature engineering is more difficult because it's domain-specific, while learners can be largely general-purpose**. Of course, one of the holy grails of machine learning is to automate more and more of the feature engineering process.



Feature engineering maps raw data to ML features. Image Credits: https://developers.google.com/machine-learning/crash-course/representation/feature-engineering

4. Learn Many Models, Not Just One

In the early days of machine learning, people tried many variations of different learners but still only used the best one. But then researchers noticed that, if instead of selecting the best variation found, we combine many variations, the results are better often much better and with only a little extra effort for the user. Creating such model ensembles is now very common:

- In the simplest technique, called **bagging**, we use the same algorithm but train it on different subsets of original data. In the end we just average answers or combine them by some voting mechanism.
- In **boosting**, learners are trained one by one sequentially. Each subsequent one paying most of its attention to data points that were mis-predicted by the previous one. And continuing until we are satisfied with the results.
- In **stacking**, the output of different independent classifiers become the input of a new classifier which gives the final predictions.

In the Netflix prize, teams from all over the world competed to build the best video recommender system. As the competition progressed, teams found that they obtained the best results by combining their learners with other teams', and merged into larger and larger teams. The winner and runner-up were

both stacked ensembles of over 100 learners and combining the two ensembles further improved the results. ***Together is better!***

5. Correlation Does Not Imply Causation

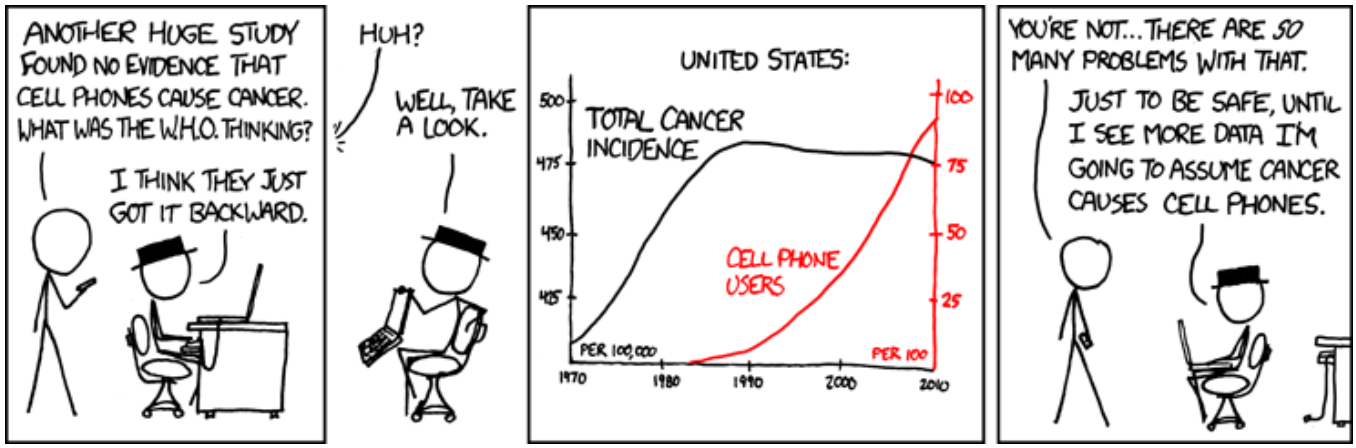


Image Credits: https://xkcd.com

We have all heard that correlation does not imply causation but still people frequently think it does.

Often the goal of learning predictive models is to use them as guides to action. If we find that beer and diapers are often bought together at the supermarket, then **perhaps** putting beer next to the diaper section will increase sales. But unless we do an actual experiment it's difficult to tell if this is true. Correlation is a **sign of a potential** causal connection, and we can use it as a **guide to further investigation** and not as our final conclusion.

> **Note**: ***This lesson is an excerpt from my blogpost on this topic. You can read the full article here.***