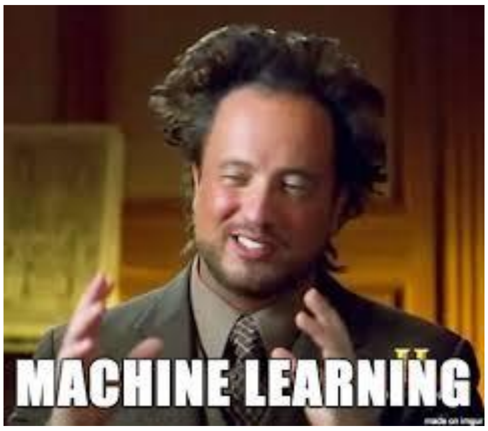
Chapter 1: 6 jars—A complete pipeline in Machine Learning

Ever wondered what machine learning is and why it means to you? Please read through the below blog post to understand what exactly machine learning is and how we put the entire machine learning lifecycle in 6 jars.

Introduction:



There are many formal definitions for machine learning in the internet. Let me quickly give a short description of what machine learning is. Just rearrange the words 'Machine learning—A learning Machine'. Be it a system or a machine or a program, it learns itself on how to react to any instance. Just imagine the instance as an unseen data. In conventional rule based program, we hardcode rules according to the different scenarios we expect in the data. But in ML, ***we feed data and the system gets programmed automatically***.

So the art of making the system to learn from the bunch of data and come up with the best approximate relationship between input and output is called machine learning. The main theme of this post is to understand how the entire ML pipeline can be put into six separate jars.

Data
Task
Model
Loss
Learning
Evaluation

6 jars in ML—6 phases in ML

Let’s take an example problem and see how different phases of this problem can be put into each jar. Given the data of an employee, the problem statement we take here is to predict if he churns out from the company or not. This kind of problem can also be solved in conventional rule based programming, but might not be a good idea. The following are the reasons for it: i) Given plentiful data, it’s hard for a human brain to break down data into so many conditions. ii) Ok, somehow we visualized the data and identified the conditions. How many possible combinations of conditions could we look up? iii) There are some rules that are in-expressive. It might be hectic for a human brain. Isn’t it? So you may need an intelligent system that finds the solution for this problem from the data we supply without needing human intervention in breaking down data into conditions. Let’s see how ML handles this task while explaining every jar/phase in parallel.

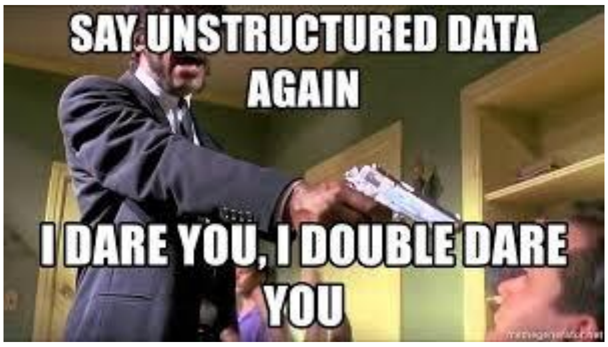
Jar 1—Data

Data is the fuel for machine learning. The availability of plenty of data makes ML outperform the conventional rule based programs. For a machine to understand the data, it must be in numeric form. The data may come in any form: Image, Text, Video or anything. It must first be converted into numeric form. First let’s see the different types of data. Data can be broadly classified into **structured/un-structured data**. Structured data, as the name implies will be in the structured format. Mostly structured data comes from database tables. Unlike structured data, un-structured data doesn’t take any standard form. Image/Text/Video data can be viewed as unstructured data. Unluckily, most of the data in the recent age is unstructured and it’s the duty of a data scientist to first structure the data at the initial stage in ML!

Social media: Major source of unstructured data

From where to get the data?

It is obvious in most of the cases that the data is project specific. But thanks to some of the open source sites where we could find data for FREE namely: data.gov.in, UCI repo, Google AI. Some MNCs handover the task of genarating the data to workers around the world. And they pay them for it!!! Some of the platforms include: amazon mechanical turk, figure eight, dataturks. Ok, for our problem(churn prediction), let's just imagine we get employee data from the company's database. Note that the data here is the historic records of employees and it has records for both churned and active employees.



Unstructured data must first be converted to structured format

Jar2—Task:

Once we have the data, we need to identify what task are we going to perform on it. A task is nothing but **a crisp definition of what is input and what's output**. An ML task can be broadly viewed as supervised/unsupervised.

Supervised learning —

If you have someone to tell what is right or what is wrong, we call it supervision. In the same manner, supervised learning has output data that tells us whether the prediction is correct or not. Supervised learning can further be divided into regression and classification.

Regression—If the output data/the data that we're going to predict is continuous, then it comes under regression problem. Say, weather forecast, sales forecasting, predicting the price of an item based on its features, etc.

Classification — If the data that we’re going to predict is categorical, it comes under this category. Say, predicting whether an employee is going to churn/not, identifying the type of tumour(malignant/benign), recognizing hand written digit(0 to 9), etc.

Unsupervised learning —

If we don’t have any benchmark to compare our prediction against or to put it in simple terms, if we don’t have any output data, we call it unsupervised learning. One of the common tasks in un-supervised learning is clustering. e.g, Amazon/Flipkart may have created several clusters and placed their customers in one of the clusters based on their shopping patterns. This helps them to provide user specific promotional offers.

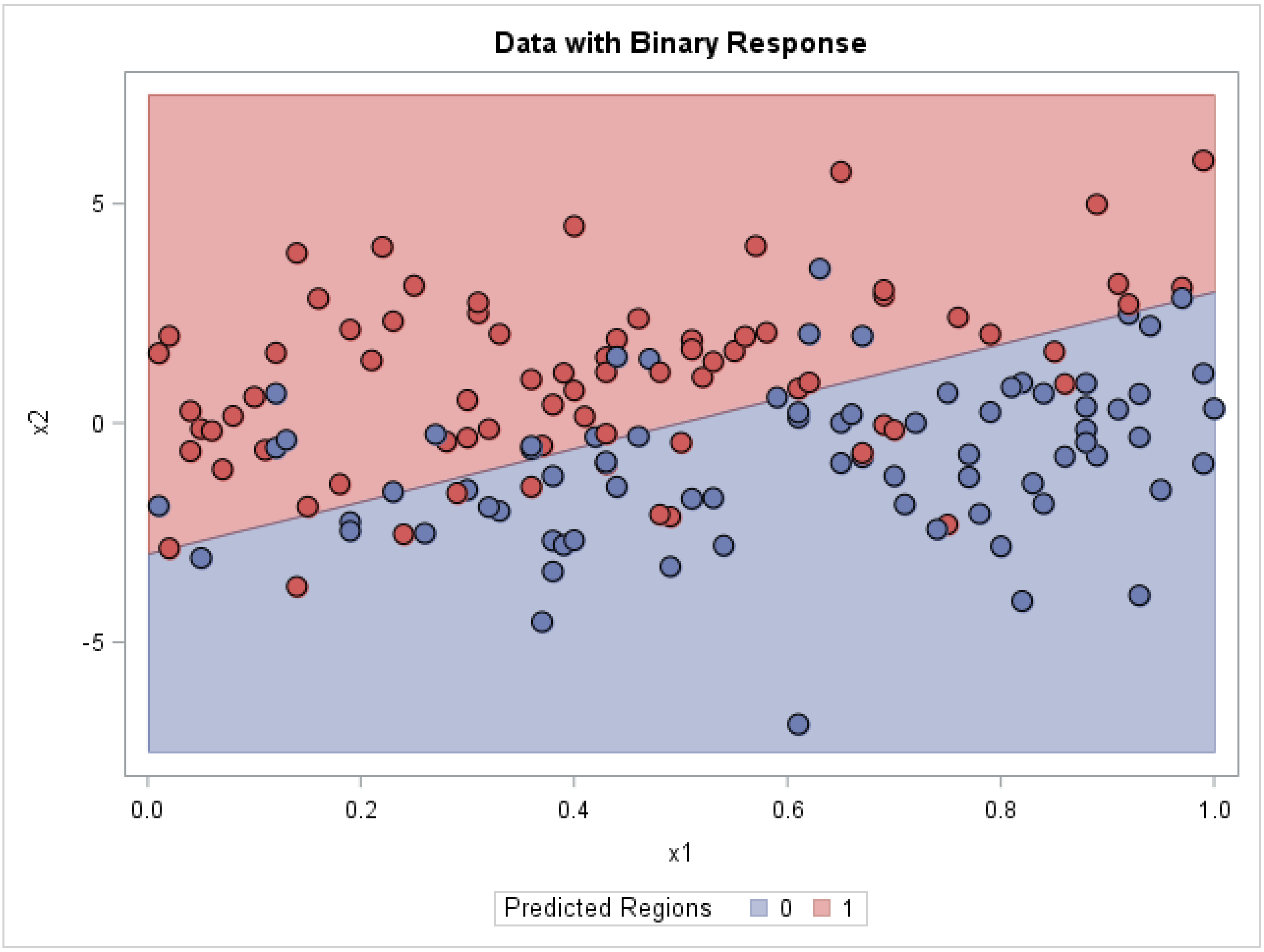
For our problem, given the employee data, the task is to predict whether he churns or not. As the output we’re going to predict is categorical(0 or 1), this is a classification problem!

Jar 3 — Model

Once we have the data and we identified what task are we going to do with it, the thing that comes next is model. Our job is to come up with a model that best approximates the relationship between input and output. A model is just a simple/complex function that best describes the relationship between our input and output.

For our problem, let’s consider our data has following features: i) no. of rewards he got in last year ii) years of experience iii) if the employee is eligible but was not promoted in last cycle(binary feature) . We call these as features and the importance of features as weights. Note that the feature importance/ weight is just a numeric value that represents how importantly that particular feature helps in predicting the output. Believe me, there is a separate field in ML called feature engineering that helps us in arriving at a best possible combinations of features for a particular problem.

We’re going to predict if he churns or not based on the above said features. Let’s consider we have around 10 lakh records of data. We call it training set.(But let’s just keep that buzzword aside).

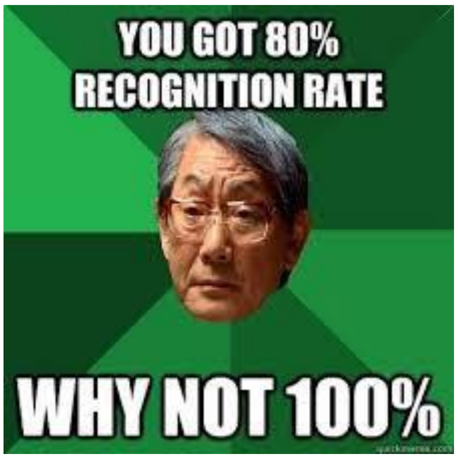


The line above represents our model that best approximates the relationship. We cannot design our model to cover exactly all the points in the training set as it may overfit the model and the model makes poor prediction on the unseen data(not getting deep into it). Given any new data, the duty of the model is to place the instance in either side of the line. Note that one region here represents he churns and the other represents he won’t. Such a simple model right? Believe me, many complex things in life have very simple solutions 😛 We may also have very complex neural network family of equation to map the relation between input and output.

Jar 4—Loss

Ok now, we have a model to tell us if the employee churns or not. But how do I know if my model makes sense in the prediction or is it just throwing out some random choice between 0 and 1. We must compare the output of the model with the actual output we have in the example data. In context of this example, the output predicted by our model for a particular example must be compared with the actual output in the data(Remember that the training data has both input and output). Now that you know the discrepancy. This is called loss. To put it in simple terms, **a loss is**

something that tells you how bad your model is. Based on the type of output(continuous/categorical), we may have different loss functions available. For continuous output, one of the common loss functions used is L2 loss, where the difference between squares of the model output and the actual output is computed as loss. Cross entropy loss is one of the common loss functions used for categorical output.



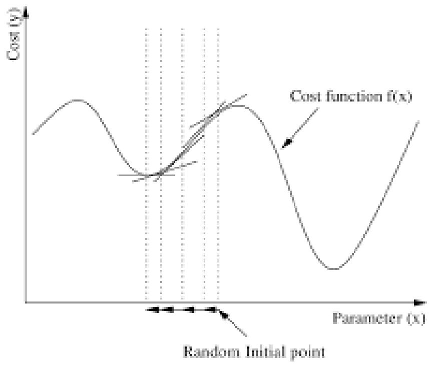
Jar 5—Learning

Given the data, given the model, given the loss function, let's now see how does our model learn the optimal values(weights—technical term for feature importance) that it can use in the function to come up with minimal loss? Here's where the computation power of system is utilized. Get hyped!

After computing the loss for a particular example, the system must come up with different set of values of the weights that minimizes the loss. But how does the system choose the new set of values? Let's see!

First let's try with things that strike in our human brain—Brute Force? Is it possible that we can try all the possible values of the weights and compute loss in all the cases and identify minimal loss? Never. How many possible combinations of values can we try in real space? So let's keep this way apart.

Optimization technique?—For what values, can a function(loss function in this case) attain its minimum. Clearly this becomes an optimization problem. Let's try to explore how optimization technique works: First we'll find the loss for a particular combination of weights.



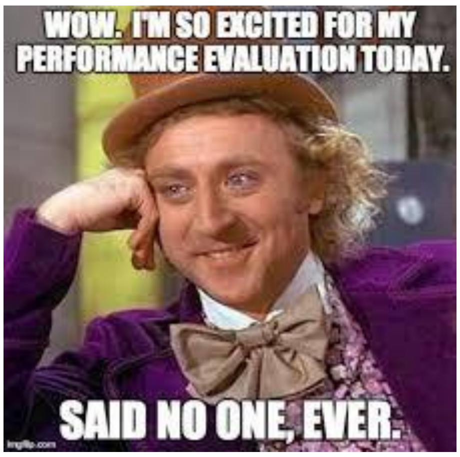
Inorder to choose the next set of values, compute derivatives of the loss function w.r to the weights and update weights accordingly.(Here comes the concept of learning rate, but let’s just skip it as it’s just an introduction post).

There are several optimization tools available that one can utilize in his ML program. Some of the common optimization techniques used are: Gradient descent optimizer, Adam optimizer, Adagrad, RMSProp.

This is the jar where the computation power of system is utilized and mostly doesn’t have human intervention. Gradient descent automatically grabs the best possible combination of values for the loss function to be minimal. Optimization algorithms work in a way that the weights of the features which are responsible for loss will be adjusted to minimize it. Nowadays we have so many libraries and built in functions for optimizations that arrive at a best optimal values for a function to be minimum.

Jar 6—Evaluation

We have now set everything. We have data, we defined the task, we defined the model and loss function and identified best possible combination of weights for our model to work better. Before we go ahead and deploy the model in production, let's first cross check how our model reacts for an unseen data. We're going to evaluate our model. The data that we use to test the model is called test data. We test our model with some unseen data and come up with the accuracy of our model to judge if it is accurate enough to be deployed?



If the model is not accurate enough, we must tweak some hyper parameters or identify some other useful combinations of features to arrive at good accuracy level.

Conclusion:

We have covered all the jars in ML and any step in ML falls in any of the listed jars. This is just an introductory post and hope it helps for an absolute beginner to understand what machine learning is and how to visualize any problem in ML.

Key takeaways from this blog post:

Machine learning outperforms in areas where conventional rule based programming fails.Unlike conventional programming, we don't need to explicitly set rules. Machine learning automatically breaks down the data into rules.The SIX jars explained above can also be seen as six phases in any machine learning project.**Enjoy learning!**