Machine Learning

The ability to automatically learn from experience and improve from experience without being explicitly programmed.

The learning process begins with observation of data in a way to look for patterns in data to make better decisions.

Literally, the purpose is to give computer the go ahead to study the pattern automatically without intervention of human.

Supervised Learning:

This is a machine learning action that maps input to output based on example. Supervised learning study the training data and produce evidence and reasoning model, which can be used for mapping new data. In supervised learning we have labelled training data.

Unsupervised Learning:

A machine learning action that draws evidence and reasoning from datasets having an input without corresponding label. The goal is to learn more about the data.

Reinforcement Learning:

In this it employs trial and error to the datasets to come up with a solution to the problem by interaction to the environment.

Back propagation:

If the outcome does not match what is hoped for, the hidden layers is adjusted. This is a concept in neural network.

Overfitting: This happened learning the datasets too well with low bias and testing on a new datasets we get high variance which negatively impacts the model ability to generalise, it can be prevented by cross validation or regularisation.

Underfitting:

This can neither model the training data nor generalise. It has poor performance both on the training data and the new datasets. It shows low variance and low bias.

Regularization:

It's used to simplify complex models that are prone to overfitting by adding penalty of absolute or square to the cost function

Ensemble Model: Uses multiple learning algorithms together for the same task to come out with the one with higher vote from prediction. It gives better accuracy (avoid error), higher consistency (avoid overfitting) and reduces bias and variance errors.

Bagging (Bootstrap aggregating): Application of multiple model of same learning algorithm trained with subsets of datasets randomly picked from the training dataset. Randomly select data points / row (one by one) into the bags (repetition allowed).

Boosting (Adaboost): it involves selecting points which gives wrong prediction and continue learning on the randomly plus the wrong selected sample and this gives a better judgement on the best prediction.

Stacking: Involves learning algorithm to combine the predictions of several other learning algorithms. Firstly, all of the other learning algorithms are trained using the available data, then a combine algorithm is trained to make a final prediction using all the predictions of other algorithms as an additional inputs. -WIKI

PAC: the main idea behind PAC is take for example we have seeds of maize taken from a bag without sorting, even though they will look the same, we suspect that there are differences, we could have three different types in one cluster but unfortunately we can't observe differences from outside. The positive correlated seeds are similar, while negative correlated are far different. Fortunately, we can plot different seeds of three dimensions, but when there are more it becomes impossible to have the plot of four to more of the dimensions. Instead, we draw principal component analysis plot. The PCA converts the correlations among all of the seeds into two dimension graphs, seeds that are highly correlated cluster together, those not are separated with a clear distance. In theme of dimension reduction we have lots of them: heat-maps, t-SNE, and Multi-dimensional scaling (MDS).

P-value: P-value and probability are related but not the same e.g. we flip a coin, fifty percent we get heads and fifty percent we get tails, when we flip the coin second time fifty percent we get heads and fifty percent we get tails. What's the probability of getting two heads in a row and what is the p-value of getting two heads in a row? In case of probability we get ¼ as two heads for possible flips of coin twice and the probability of getting two tails is just as before ¼, the probability of getting one heads and one tails is 2/4. We got a probability for two heads which is ¼ and for p-value we got a probability of two heads ¼ plus two tails ¼ which gives 0.5 plus unusual probability which comes as zero, we treat the outcome of getting head and tail the same because other doesn't matter.

Decision Trees

A decision tree ask a question and classifies base on the answer. The simplest decision tree ask simple question like if a person has money to give to needy, it's either a needy gets a share or no share of the money. A classification can be categorical or numeric. The numeric shows a chicken weighs between 15 and 20 grams. If decision is made, it splits into true and false. The true weight lies in the direction of true arrow while the false weight lies in the direction of the false and this gives us the prediction needed to decide where our answer lies. Sometimes we could have a complicated trees that have both numeric and classification data. Illustration of complicated tree is of this form where you have from the root a decision to make of a woman having an age of 50 and above followed by split to both numerical and classification where it will have resting heart rate on one side and amount of sugar in take on the other, it doesn't have to be same question on both left and right, on the either side the classification question can be ask first followed by numeric while on the other side the numeric could be ask first followed by classification, it all starts from the top until you go down the tree where you get to the point where you can't go any further and this is how samples are classified.

Let's talk in detail about the tree structure, it grows downward, the root or root node at the top having arrows pointing away from it without any pointing into it, followed by the internal nodes or the nodes having arrows pointing into them and pointing away from them, then the leaves having arrows pointing into them without arrow pointing away from them.

Now we can talk about how to go from a raw table data to a decision tree. We would like to create a tree that uses clarity, decisiveness, courage, passion, humility to predict whether a leader is a great one. The first thing we would want to know is whether clarity, decisiveness, courage, passion or humility should be at the very top of our tree. We start by looking at clarity alone to predict a great leader in this we have two branches false and true, in the true leaf we have yes and no and the false leaf yes and no as well, if the clarity feature has yes and a great leader has yes we give it to the true arrow away from the clarity and no great leader will end under the no but still true arrow if the clarity is yes, while the no clarity will end up on the false arrow under yes and no depend on the great leader value. Now we do the exact same thing for all the features, our goal is to decide whether clarity, decisiveness, courage, passion or humility should be the first in our decision tree. The total number of a great leader is different from each of the feature because some leaders have measurement for clarity not for decisiveness etc. Because none of the leaf nodes are 100% yes great leader or 100% no great leader, they are all considered impure

To determine which separation is the best we need a way to measure and compare impurity. There are bunch of ways to measure impurity but I am going to focus on this popular one called GINI. Let's focus on calculating GINI impurity

Impurity = 1 – (the probability of yes)^2 - (the probability of NO)^2

After calculating the GINI impurity for both leaf nodes we can calculate the total GINI impurity for using Clarity to separate leader either great or none great.

We take the total of the leave on the left side and divide by the total of the leave for both leaf node and multiply by the GINI impurity obtained with the left leaf node and the same applied to the right leaf node both are added together to obtained a value. All feature are calculated same way and the values are compared, the one with minimum value is use as the top most separator, and it's followed same way to separate until we could no longer separate.

But what about with numeric data, imagine we have data table of the form first column height followed by obesity, each height value has it corresponding obesity value either yes or no. how can we determine what the best height to divide the patient? The patient initially are sorted by height lowest to highest, calculate the average height for all the adjacent values, and calculate the impurity value for all the average weight. Take the adjacent average to make a separation that gives either less than sign to give us where the upper or lower lies to calculate the GINI impurity value, and use to multiply corresponding total probability from each separation. We will calculate the impurity value for the other weight as well, the lowest impurity value will be the first separation.

We are going to talk about rank data like rank my picture on a scale of 1 to 5 and multiple choice question like which colour do you like blue, red, green or blue? On this we calculate impurity scores for all of the possible ranks. So, if people rank my jokes for 1 to 4 and four being the funniest we can calculate all other impurity below without including the 4^{th} because including 4 that will include everyone.

In case of multiple choice we will calculate impurity on each one as well as possible combination like having blue or red, red or green, etc.

Decision trees are easy to build, easy to use and easy to interpret, but in practice they are not that awesome because of inaccuracy. They work well with data use to create them, but they are not flexible when it comes to classifying them.

Random Forest

Random forest are made of decision trees. Random forest combine the simplicity of decision trees and flexibility resulting a vast improvement in accuracy. Firstly, bootstrapped dataset is created. Imagine you have original dataset and bootstrapped dataset is created randomly with sample from original set including allowed picking same sample more than once. Create a decision tree using the bootstrapped dataset, but only use random features at each step. Make more bootstrapped dataset and build more decision trees, you can build many decision trees as possible. This varieties make random forest much effective than decision tree. In this we run samples over all the trees we made to find which has more number of Yes or No. Bootstrapping the data plus using the aggregate to making decision is called Bagging. We can also have out of bag dataset that is when we do not include it in the bootstrap which are entries which do not make bootstrap dataset, since it didn't make the bag we can run it to see if it correctly classify the sample as great leader

Linear Regression

The simple Linear Regression is given by Y = mx + c known as best fit line

m = slope which stands for every change in x-axis what will be the change in y-axis

c = intercept indicate when x = 0 what will be the position of Y

Having the horizontal axis and the vertical axis we can determine our observation and prediction, suppose I have two features with respect to sizes my prices would be determined for any size of a house involved. Let's have scattered points in space between y-axis and x-axis and a best fit line drawn which stands a model, whenever we have an extension on the x-axis we could still determine the y-axis which means for any new point on the x-axis it will give us a y-axis matching. The best fit line do not pass through all the observation points as it passes through point that line is called predicted point, the predicted point on the best fit line subtract from the observation points give us what we call error, the most important point is to minimise the error by choosing the best fit line, it's calculated as below which leads to the minimum cost function

Y = observation points

$\bar{Y}$ = predicted points on best fit line

n = number of observation points

Cost Function = $\frac{1}{2n}\sum_{i=1}^{n}(\bar{Y} - Y)^2$

The next most important part is the slope (x-axis) vs the cost function (y-axis), that shows for every slope initialise what will be the cost function which gives rise to concave curve called gradient descent. If you get this gradient descent, when should you know to select m (slope) value which look good for the best fit line? Our next point is to get the global minimum on the curve which is the lowest part of the curve this gives the best slope that determine the best cost function. It's calculated by taking derivative of a slope multiply by learning rate with subtraction from given slope, the learning rate must be a small value to avoid a bigger steps which could deprive it reaching a global minimum.

m = m - $\frac{\partial m}{\partial m}\alpha$

α = Learning rate

m = slope

Logistic Regression

It's similar to linear regression except it predict something that is true or false instead of predicting something continuous, instead of fitting line to data, it fits squiggle line to data as s shaped or sigmoid, the curve goes from zero to one. The major difference between logistic regression and linear regression is how the line fits to the data. It use something called maximum likelihood instead of least square or residual in linear regression to calculate all the probability scaled by X-axis of observing a Y-axis points, s shaped gets shift until the curve with the maximum likelihood is obtained. On the Y-axis is confined to the probability between zero and one. The dependant variable on the Y-axis are categorical opposite to that of linear regression with continuous variables. In getting a result if is close to zero or one we set up a threshold value, below the threshold its zero, above the threshold it's one. The Y-axis are probability of events to happen which you are trying to predict, the X-axis are the independent event variables which determine the occurrence of an events dependant Y-axis, C is the constant which will be probability of events happening when no other factors are considered.

$\log\frac{Y}{1-Y} = C + B_1X_1 + B_2X_2 + \ldots.$

Y = confined probability

Support vector machine

Let's start by imagine we measure the mass of bunch of mice on a straight horizontal line. The red dots represent the mass that are not obese and the green dots represent mice that are obese base on this observation we can pick a threshold then we get a new observation that get a less mass than the threshold we can classify it as not obese, and when we get new observation more than the threshold we classify we can classify as obese. However, what about if we get a new observation slightly part on the threshold and part away from the threshold still close to not obese? And because is little more than the threshold we classify as obese but this doesn't make sense since is close to the observation not obese. This threshold is pretty weak. We can do better than this. Going back to the observation of the datasets we can focus on the edges of the observations and use the midpoint between them as the threshold. Now when new observation falls on the left side of the observation it will be closer to the observation that are not obese than to the obese observations. So, it make sense to classify this new observation as not obese. The shortest distance between observation and the threshold is called margin, since we put the threshold between these two observations the distance between these observations and threshold are the same and both reflect the margin, when the threshold is large between the two observations the margin can be large as it can be. When we use the threshold that gives us the largest margin to make classification we are using maximal classifier, it looks pretty cool but what about we have not obese very close to the obese an outlier? In this case the maximal margin classifier

will be so much close to the obese observations and very far from the majority that are not obese even though we have observation not obese close to the obese we still classify as not obese this is because maximal margin outlier are super sensitive to outlier in the training data and made them pretty lame. To make a threshold that are not so sensitive to outlier we must allow misclassifications, for example if we but the threshold in the middle we will misclassify the observation not obese close to the obese observations. However, if we get a new observation that is below the outlier but obese that make sense because it is closer to the obese observation. Choosing a threshold that allow misclassifications is an example of the Bias/Variance trade-off that plague all the machine learning. Before we allow misclassifications we will pick a threshold that was very sensitive to the training data (low bias) and it perform poorly when we got new data (high variance). In contrast, when we pick a threshold that is less sensitive to the training data and allow misclassifications (high bias) It perform better because it has low variance. When we allow misclassification the distance between the observation and the margin is called soft margin. The name support vector classifier comes from the fact that the observations on the edge and within the soft margin are called support vectors. When the data are one dimensional, the support vector is a point on a one dimensional number line. In mathematical jargon a point is a flat affine 0-dimensional subspace. When the data are in two dimension, the support vector classifier is a one dimensional line in two dimensional space. In mathematical jargon a line is a flat affine one dimensional subspace. When the data are three dimensional, the support vector classifier is a two dimensional plane in a three dimensional space. In mathematical jargon a plane is a flat affine two dimensional subspace, and when the data are in four or more dimensions, the support vector classifier is a hyperplane, in mathematical jargon a hyperplane is a flat affine subspace. Technical speaking, all flat affine subspaces are called hyperplanes. Technical speaking, this one dimensional line is called hyperplane. Support vector classifier can handle outliers, they can allow misclassifications, and they can handle overlapping misclassifications. What about if we have drugs which cure obesity and surrounded by non-cure drugs, if the lower ones are not able to cure and the upper ones too accept the middle ones? Since maximal margin classifier and support vector classifier are good at solving this, now let's introduce support vector machine, we start by adding a Y-axis to the already observed dosage and the Y-axis will be square of the dosage. Since each observation has Y and X-axis coordinates, the data are now two dimensional. Now that the data are two dimensional we can draw support vector classifier to separate people who are cure and those that are not cured and the support vector classifier can be used for new observations, it classify the observation not cure because it ended up on the other side of the support vector classifier and leaving the cure dosage on the other side. If we get a new observation we will square the dosage and found out where it falls to. The real idea about support vector machine is to start with data in relatively low dimension like with one dimension, move the data into higher dimension, find the support vector classifier that separate the higher dimensional data into two groups. Kernel polynomial uses degree of polynomial to create support vector classifier

K-means Clustering

Imagine you had a data that you could plot in a line, and you knew you needed to put it into three clusters. Maybe they are measurements from three different types of tumours or other cell types. Let's make a three relative clear clusters. But, rather than relying on our eyes, let see if we can get a computer to identify the same three clusters. To do this we use k-means clustering. Data that hasn't been cluster. Firstly, select the number of cluster you want to identify in your data this is the k in k-means clustering. Let's assume we select k = 3 this is to say we want to identify 3 clusters. Second, randomly select three distinct data point. Measure the distance between the first point and three initial clusters. Assign the first point to the nearest cluster, do the same thing with the next point. Assign the point to the nearest cluster, we figure out the third point which cluster it belongs to by measuring the distances, we assign the point to the nearest cluster. Now that we got all the point to each cluster we will calculate the mean of the cluster. Then we repeat what we just did by measuring and clustering using the mean values. The best way is to keep track of the data and their total variance, and do the whole thing over again. Calculate the mean of each cluster and re-cluster base on new means. Repeat until cluster no longer change. If the data are cluster we will sum the variation within each cluster and we do it all again. How to figure out what k value to use? The way to decide is to try different value for k. Each time we add a new cluster the total variation become smaller

K-nearest neighbour

Start with datasets with known categories. Let's take for example we have different colours of maize. We will cluster the data, we will use PCA (principal component analysis). We add a new cell, with unknown category, to the PCA plot. We don't know this maize category because it was taken from another maize bag where they are not properly sorted. What we do is that we want to classify this new maize. We classify the new maize by looking at the nearest neighbour. If the K in the nearest neighbour is equal to one, then we will only use the nearest neighbour to define the category. If k=10 we will use the nearest neighbours. If k =10 and the new maize is between two or more categories, we simply pick the category that gets the most votes. There is no biological or physical way to determine the best value for K, so you may have to try a few value before settling on one.

Naïve Bayes

Multinomial Naïve Bayes Classifier: imagine we receive normal messages from families and friends and also receive spam (unwanted messages that are usually scams or unsolicited advertisements) and we wanted to filter out the spam messages. So, the first thing we do is to make a histogram of all words that occur in the normal messages from

friends and family. We can use the histogram to calculate the probabilities of seeing each word, given that it was in a normal message. For example the probability of seeing a word dear P (Dear|Normal) given that it is seeing in the normal message which is the total number of dear that occur over the total number of all words in the normal messages. We also make histogram all the words that appear in spam and calculate the probability of seeing a word dear given that we saw in a spam messages likewise we calculate the probability of remaining words given they are all in the spam. Because we have calculated probabilities of discrete not continues like weight or height, the probabilities are also called likelihoods. Probabilities or likelihoods are interchangeable so don't sweat it. Probabilities are the areas under a fixed distribution, while likelihoods are the Y-axis values for fixed data points with distribution that can be moved. Now imagine we got a new message Dear Friend, we will start with an initial guess about the probability that any message, regardless of what it says, is a normal message P(N) which is called prior probability from the total normal occurrence over the normal plus spam. Now we multiply the initial guess with the probability that the dear occurs in a normal message and the probabilities that other words occur in the normal message. However, it is proportional to the probability that the message is normal, given that it says dear friend, we do another regardless of what it says it is spam, just like before the guess can be any probability that we want and we do the maths such as before. And if the score we got from normal is greater than the score we got from spam we will decide that the word dear friend is a normal message.

Gaussian Naïve Bayes Classifier: imagine we wanted to predict if someone would love matches played by Manchester city in 2017 fixture or not, we collected data from people that love the matches and those who do not, we measure the amount of water drank each fixture, how much popcorn and how much candy they ate. Take the mean of water for people who love to watch the matches, standard deviation to observe the normal distribution likewise that of those who do not love the fixtures is taken to observe the normal distribution, it follows all the attributes and the corresponding Gaussian distribution. We take the initial guesses from love and not love P (Love) and P (not love) called prior probabilities times the likelihood that they drink water given that they love, times the next attributes, if we get a very small likelihood, it's good we take the log of the value to prevent underflow to avoid running to zero which result error. So we take the log of every value to get a negative value, we do same with over those that do not love Manchester city matches times the likelihood that they drink given not love matches times following attributes and take log of every values and we get negative value. So, the scores compare with the two part the bigger will assume compromise