The universal workflow of machine learning

Defining the problem and assembling a dataset

First, you must define the problem at hand: What will your input data be? What are you trying to predict? You can only learn to predict something if you have available training data: What type of problem are you facing? Is it binary classification? Multiclass classification? Scalar regression? Vector regression? Multiclass, multilabel classification?

Choosing a measure of success

To control something, you need to be able to observe it. To achieve success, you must define what you mean by success—accuracy? Precision and recall? Customer-retention rate? Your metric for success will guide the choice of a loss function: what your model will optimize. It should directly align with your higher-level goals, such as the success of your business.

Deciding on an evaluation protocol

Once you know what you're aiming for, you must establish how you'll measure your current progress.

Preparing your data

Once you know what you're training on, what you're optimizing for, and how to evaluate your approach, you're almost ready to begin training models. But first, you should format your data in a way that can be fed into a machine-learning model.

Developing a model that does better than a baseline

Your goal at this stage is to achieve *statistical power:* that is, to develop a small model that is capable of beating a dumb baseline.

Scaling up: developing a model that overfits

Once you've obtained a model that has statistical power, the question becomes, is your model sufficiently powerful?

The law of large numbers

As the number of repetitions of a probability experiment increases, the proportion with which a certain outcome is observed gets closer to the probability of the outcomes.

Tests for heteroscedasticity

In statistics, a collection of random variables is **heteroscedastic** if there are sub-populations that have different variabilities from others. Here "variability" could be quantified by the variance or any other measure of statistical dispersion.

Supervised learning is simply a process of learning algorithm from the training dataset. Supervised learning is where you have input variables and an output variable, and you use an algorithm to learn the mapping function from the input to the output.

Unsupervised learning is modeling the underlying or hidden structure or distribution in the data in order to learn more about the data. Unsupervised learning is where you only have input data and no corresponding output variables.

F1 score (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision *p* and the recall *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results returned by the classifier, and *r* is the number of correct positive results divided by the number of all relevant samples.

Confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Once you understand these four parameters then we can calculate Accuracy, Precision, Recall and F1 score.

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have

symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.

Accuracy = TP+TN/TP+FP+FN+TN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

P value is a number, calculated from a statistical test, that describes how likely you are to have found a particular set of observations if the null hypothesis were true. P values are used in hypothesis testing to help decide whether to reject the null hypothesis.

The smaller the p value, the more likely you are to reject the null hypothesis.

T test is a statistical test that is used to compare the means of two groups. It is often used in hypothesis testing to determine whether a process or treatment actually has an effect on the population of interest, or whether two groups are different from one another.

Type I error means rejecting the null hypothesis when it's true. It means concluding that results are statistically significant when they came about purely by chance or because of unrelated factors.

Type II error means not rejecting the null hypothesis when it's actually false. This is not quite the same as "accepting" the null hypothesis, because hypothesis testing can only tell you whether to reject the null hypothesis. Instead, a Type II error means failing to conclude there was an effect when there actually was. In reality, your study may not have had enough statistical power to detect an effect of a certain size.

In a normal distribution, data is symmetrically distributed with no skew. When plotted on a graph, the data follows a bell shape, with most values clustering around a central region and tapering off as they go further away from the center.

Normal distributions are also called Gaussian distributions or bell curves because of their shape. The standard normal distribution, also called the z-distribution, is a special normal distribution where the mean is 0 and the standard deviation is 1. Any normal distribution can be standardized by converting its values into z scores. Z scores tell you how many standard deviations from the mean each value lies.

Poisson distribution is a discrete probability distribution. It gives the probability of an event happening a certain number of times (k) within a given interval of time or space.

Poisson distribution has only one parameter, λ (lambda), which is the mean number of events. The graph below shows examples of Poisson distributions with different values of λ.

T-distribution, also known as Student's t-distribution, is a way of describing data that follow a bell curve when plotted on a graph, with the greatest number of observations close to the mean and fewer observations in the tails. It is a type of normal distribution used for smaller sample sizes, where the variance in the data is unknown.

Chi-square (X2) distribution is a continuous probability distribution that is used in many hypothesis tests. The shape of a chi-square distribution is determined by the parameter k.

NLP

Natural language processing is an area of research in computer science and artificial intelligence (AI) concerned with processing natural languages such as English or Mandarin. This processing generally involves translating natural language into data (numbers) that a computer can use to learn about the world. And this understanding of the world is sometimes used to generate natural language text that reflects that understanding.

Stemming – grouping the various inflections of a word into the same bucket or cluster, based only on their spelling. For example, you can stem the word "running" to "run."

Tokenization – In NLP, tokenization is a kind of document segmentation. Segmentation breaks up text into smaller chunks or segments, with more focused information, content. Segmentation can include breaking a document into paragraphs, paragraphs into sentences, sentences into phrases, or phrases into tokens (usually words) and punctuation.

Tokenization is the first step in an NLP pipeline, so it can have a big impact on the rest of your pipeline. A tokenizer breaks unstructured data, natural language text, into chunks of information that can be counted as discrete elements.

N-gram – An n-gram is a sequence containing up to n elements that have been extracted from a sequence of those elements, usually a string. In general, the "elements" of an n-gram can be characters, syllables, word, or even symbols like "A", "T", "G", and "C" used to represent a DNA sequence.

Stop words – Stop words are common words in any language that occur with a high frequency but carry much less substantive information about the meaning of a phrase. Historically, stop words have been excluded from NLP pipelines to reduce the computational effort to extract information from a text. Even though the words themselves carry little information, the stop words can provide important relational information as part of an n-gram.

Bags of words – are vectors of word counts or frequencies.

TF-IDF – stands for term frequency times inverse document frequency. Term frequencies are the counts of each word in a document. Inverse document frequency means that you'll divide each of those word counts by the number of documents in which the word occurs.

LEMMATIZATION

If you have access to information about connections between the meanings of various words, you might be able to associate several words together even if their spelling is quite different. This more extensive normalization down to the semantic root of a word—its lemma—is called lemmatization.

Word embedding is a representation of a word. Embedding is used in text analysis. Typically, the representation is a real-valued vector that encodes the meaning of the word in such a way that words that are closer in the vector space are expected to be similar in meaning.
Word embeddings can be obtained using language modeling and feature learning techniques, where words or phrases from the vocabulary are mapped to vectors of real numbers.

Transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. It is used primarily in the fields of natural language processing (NLP) and computer vision (CV).

Like recurrent neural networks (RNNs), transformers are designed to process sequential input data, such as natural language, with applications towards tasks such as translation and text summarization. However, unlike RNNs, transformers process the entire input all at once.

The **attention mechanism** provides context for any position in the input sequence. For example, if the input data is a natural language sentence, the transformer does not have to process one word at a time. This allows for more parallelization than RNNs and therefore reduces training times.