Comparison of common ML Algorithms -

Naïve Bayes	SVM	Logistic Regression	Random Forest	Decision Trees
• Simple to implement • Computationally fast • Works well with high dimensions • Relies on independence assumption	• Performs well when variables are linearly separated • Performs well with non-linear boundary • Can handle high dimensional data well • Susceptible to overfitting / training issues depending on kernel	• Low variance • Provides probabilities for outcomes • Works well with diagonal (feature) decision boundaries • High bias	• Important when dealing with multiple features which may be correlated • Reduced variance • Not as easy to visually interpret	• Easy to interpret visually when the trees only contain several levels • Can easily handle qualitative features • Works well with decision boundaries parallel to the feature axis • Prone to overfitting

Pros & Cons of common ML Algorithms -

Algorithm	Best at	Pros	Cons
Random Forest	Apt at almost any machine learning problem Bioinformatics	Can work in parallel Seldom overfits Automatically handles missing values No need to transform any variable No need to tweak parameters Can be used by almost anyone with excellent results	Difficult to interpret Weaker on regression when estimating values at the extremities of the distribution of response values Biased in multiclass problems toward more frequent classes
Linear regression	Baseline predictions Econometric predictions Modelling marketing responses	Simple to understand and explain It seldom overfits Using L1 & L2 regularization is effective in feature selection Fast to train Easy to train on big data thanks to its stochastic version	You have to work hard to make it fit nonlinear functions Can suffer from outliers

Support Vector Machines	Character recognition Image recognition Text classification	Automatic nonlinear feature creation Can approximate complex nonlinear functions	Difficult to interpret when applying nonlinear kernels Suffers from too many examples, after 10,000 examples it starts taking too long to train
K-nearest Neighbors	Computer vision Multilabel tagging Recommender systems Spell checking problems	Fast, lazy training Can naturally handle extreme multiclass problems (like tagging text)	Slow and cumbersome in the predicting phase Can fail to predict correctly due to the curse of dimensionality
Naive Bayes	Face recognition Sentiment analysis Spam detection Text classification	Easy and fast to implement, doesn't require too much memory and can be used for online learning Easy to understand Takes into account prior knowledge	Strong and unrealistic feature independence assumptions Fails estimating rare occurrences Suffers from irrelevant features
Logistic regression	Ordering results by probability Modelling marketing responses	Simple to understand and explain It seldom overfits The best algorithm for predicting probabilities of an event Fast to train Easy to train on big data	You have to work hard to make it fit nonlinear functions Can suffer from outliers
K-means	Segmentation	Fast in finding clusters Can detect outliers in multiple dimensions	Suffers from multicollinearity Clusters are spherical, can't detect groups of other shape Unstable solutions, depends on initialization

Approach to a Machine Learning problem depending on the end goal -

1. Regression

Regression is the supervised learning task for modeling and predicting continuous, numeric variables. Examples include predicting real-estate prices, stock price movements, or student test scores.

Regression tasks are characterized by labelled datasets that have a numeric target variable. In other words, you have some "ground truth" value for each observation that you can use to supervise your algorithm.

Linear Regression

1.1. (Regularized) Linear Regression

- Linear regression is one of the most common algorithms for the regression task. In its simplest form, it attempts to fit a straight hyperplane to your dataset (i.e. a straight line when you only have 2 variables). As you might guess, it works well when there are linear relationships between the variables in your dataset.
- **Strengths:** Linear regression is straightforward to understand and explain, and can be regularized to avoid overfitting. In addition, linear models can be updated easily with new data using stochastic gradient descent.
- **Weaknesses:** Linear regression performs poorly when there are non-linear relationships. They are not naturally flexible enough to capture more complex patterns, and adding the right interaction terms or polynomials can be tricky and time-consuming.

1.2. Decision Tree

Regression trees (a.k.a. decision trees) learn in a hierarchical fashion by repeatedly splitting your dataset into separate branches that maximize the information gain of each split. This branching structure allows regression trees to naturally learn non-linear relationships.

Ensemble methods, such as Random Forests (RF) and Gradient Boosted Trees (GBM), combine predictions from many individual trees. We won't go into their underlying mechanics here, but in practice, RF's often perform very well out-of-the-box while GBM's are harder to tune but tend to have higher performance ceilings.

- **Strengths:** Decision trees can learn non-linear relationships, and are fairly robust to outliers. Ensembles perform very well in practice, winning many classical (i.e. non-deep-learning) machine learning competitions.
- **Weaknesses:** Unconstrained, individual trees are prone to overfitting because they can keep branching until they memorize the training data. However, this can be alleviated by using ensembles.

1.3. Deep Learning

Deep learning refers to multi-layer neural networks that can learn extremely complex patterns. They use "hidden layers" between inputs and outputs in order to model intermediary representations of the data that other algorithms cannot easily learn.

They have several important mechanisms, such as convolutions and drop-out, that allows them to efficiently learn from high-dimensional data. However, deep learning still requires much more data to train compared to other algorithms because the models have orders of magnitudes more parameters to estimate.

- **Strengths:** Deep learning is the current state-of-the-art for certain domains, such as computer vision and speech recognition. Deep neural networks perform very well on image, audio, and text data, and they can be easily updated with new data using batch propagation. Their architectures (i.e. number and structure of layers) can be adapted to many types of problems, and their hidden layers reduce the need for feature engineering.
- **Weaknesses:** Deep learning algorithms are usually not suitable as general-purpose algorithms because they require a very large amount of data. In fact, they are usually outperformed by tree ensembles for classical machine learning problems. In addition, they are computationally intensive to train, and they require much more expertise to tune (i.e. set the architecture and hyperparameters).

1.4. Nearest Neighbors

Nearest neighbors algorithms are "instance-based," which means that that save each training observation. They then make predictions for new observations by searching for the most similar training observations and pooling their values.

These algorithms are memory-intensive, perform poorly for high-dimensional data, and require a meaningful distance function to calculate similarity. In practice, training regularized regression or tree ensembles are almost always better uses of your time.

2. Classification

Classification is the supervised learning task for modeling and predicting **categorical** variables. Examples include predicting employee churn, email spam, financial fraud, or student letter grades.

As you'll see, many regression algorithms have classification counterparts. The algorithms are adapted to predict a class (or class probabilities) instead of real numbers.

Logistic Regression

2.1. (Regularized) Logistic Regression

Logistic regression is the classification counterpart to linear regression. Predictions are mapped to be between 0 and 1 through the logistic function, which means that predictions can be interpreted as class probabilities.

The models themselves are still "linear," so they work well when your classes are linearly separable (i.e. they can be separated by a single decision surface). Logistic regression can also be regularized by penalizing coefficients with a tuneable penalty strength.

- **Strengths:** Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting. Logistic models can be updated easily with new data using stochastic gradient descent.
- **Weaknesses:** Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

2.2. Classification Tree (Ensembles)

Classification trees are the classification counterparts to regression trees. They are both commonly referred to as "decision trees" or by the umbrella term "classification and regression trees (CART)."

- **Strengths:** As with regression, classification tree ensembles also perform very well in practice. They are robust to outliers, scalable, and able to naturally model non-linear decision boundaries thanks to their hierarchical structure.
- **Weaknesses:** Unconstrained, individual trees are prone to overfitting, but this can be alleviated by ensemble methods.

2.3. Deep Learning

To continue the trend, deep learning is also easily adapted to classification problems. In fact, classification is often the more common use of deep learning, such as in image classification.

- **Strengths:** Deep learning performs very well when classifying for audio, text, and image data.
- **Weaknesses:** As with regression, deep neural networks require very large amounts of data to train, so it's not treated as a general-purpose algorithm.

2.4. Support Vector Machines

Support vector machines (SVM) use a mechanism called kernels, which essentially calculate distance between two observations. The SVM algorithm then finds a decision boundary that maximizes the distance between the closest members of separate classes.

For example, an SVM with a linear kernel is similar to logistic regression. Therefore, in practice, the benefit of SVM's typically comes from using non-linear kernels to model non-linear decision boundaries.

- **Strengths:** SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.
- **Weaknesses:** However, SVM's are memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets. Currently in the industry, random forests are usually preferred over SVM's.

2.5. Naive Bayes

Naive Bayes (NB) is a very simple algorithm based around conditional probability and counting. Essentially, your model is actually a probability table that gets updated through your training data. To predict a new observation, you'd simply "look up" the class probabilities in your "probability table" based on its feature values. It's called "naive" because its core assumption of conditional independence (i.e. all input features are independent from one another) rarely holds true in the real world.

- **Strengths:** Even though the conditional independence assumption rarely holds true, NB models actually perform surprisingly well in practice, especially for how simple they are. They are easy to implement and can scale with your dataset.
- **Weaknesses:** Due to their sheer simplicity, NB models are often beaten by models properly trained and tuned using the previous algorithms listed.

3. Clustering

Clustering is an unsupervised learning task for finding natural groupings of observations (i.e. clusters) based on the inherent structure within your dataset. Examples include customer segmentation, grouping similar items in e-commerce, and social network analysis.

Because clustering is unsupervised (i.e. there's no "right answer"), data visualization is usually used to evaluate results. If there is a "right answer" (i.e. you have pre-labelled clusters in your training set), then classification algorithms are typically more appropriate.

K-Means

K-Means is a general-purpose algorithm that makes clusters based on geometric distances (i.e. distance on a coordinate plane) between points. The clusters are grouped around centroids, causing them to be globular and have similar sizes.

This is the recommended algorithm for beginners because it's simple, yet flexible enough to get reasonable results for most problems.

- **Strengths:** K-Means is hands-down the most popular clustering algorithm because it's fast, simple, and surprisingly flexible if you pre-process your data and engineer useful features.
- **Weaknesses:** The user must specify the number of clusters, which won't always be easy to do. In addition, if the true underlying clusters in your data are not globular, then K-Means will produce poor clusters.