**Different Algorithims in** **ML**

1. Start with your goal - the list of algorithms in the question seems to point towards classification (supervised learning).

Do you just need the best classification in terms of accuracy?

Or do you also need meaningful features?

In the latter case, decision trees and Naive Bayes might be better than other, more "black box", algorithms, like SVM or ANN.

1. First of all, look at the data. Clean it. Choose appropriate features. The choice of algorithm is secondary to having lots of quality data.
2. Set up in advance criteria for success - what is your error measure? Is 75% precision good enough?
3. Resources - some algorithms need more memory or more runtime than others - e.g**. random forests need more memory**.
4. Online/batch - How soon do you need to classify a new instance? Most algorithms require intensive training, but allow for a relatively quick labeling of a new instance.
5. Scaling - for millions of instances, it is probably better to use an algorithm that scales well (e.g. **Naive Bayes**).
6. Cost of labels - you need to get the labels for the training set from somewhere. Getting labels is usually costly in time, effort and money. Some algorithms can do with less labeled data. I second[Ganesh Parameswaran](https://www.quora.com/profile/Ganesh-Parameswaran)'s recommendation for semi-supervised models, as they can do more with the same amount of labels.
7. Ensembles - In many cases, using several algorithms and combining the results (say using a majority vote) works better than individual algorithms.
8. Iterate - If you have reached your criteria for success - stop. Otherwise, try a different feature/algorithm combination.

**Supervised:**

* k-nearest neighbor
* Apriori algorithm
* Eclat algorithm
* Naive Bayes
* Bayesian networks
* Hidden Markov models
* Case-based reasoning
* Gaussian process regression
* Gene expression programming
* Support vector machines
* Random Forests
* Nearest Neighbor Algorithm

**1. Decision Trees:**You can use decision trees when you have a linear decision boundary. An example would be classifying people on the basis of their IQ:  
Over 140 - Genius  
Over 120 - Above Average  
An advantage that can attribute some meaning to the decisions. **2. Support Vector Machines:**Support Vector Machines are primarily for **binary classification**. (They can be indirectly used for multi-class classification, but I don't have much idea about it. I have done **face recognition** with SVM, but describing itwill get too technical.) SVMs have the advantage (compared to decision trees) that you can use them for non-linear decision boundaries. But the disadvantage is that you can't attribute meaning (like why the features being x result in the sample being classified y). SVMs also take a really long time to train. **3. ANN: Artificial Neural Networks**ANNs can be used for multi-class classification and non-linear boundaries. But a problem with ANN is that you have to empirically try out different topologies to arrive at the best one. Also, you generally need a huge number of training samples (lot more than what you would require in SVM) to achieve good accuracy. So, training takes a long time. You also can get probabilistic estimates of a sample belonging to a class. (Scikit-learn's function for SVM generates probabilities, but I am not sure how they do that. They probably use the distance from the decision boundary or something, but ANN directly gives you probabilities.) **4. Random Forests:**Random forests can be used for both **classification** and **regression**. You have a bunch of decision trees, each formed with a (not necessarily disjoint) subset of the features. Each tree gives a vote for the class. The sample is classified as the class with the most number of votes. Random forests can also give you an indication of feature importance. It is one of the most popular classifiers, so you should probably try it almost always.  
**5. Linear Regression\ Ridge regression:**  
Use when you want to predict continuous values, instead of classifying. Regression can be used for traffic prediction, for instance. Ridge regression reduces the variance in your predictions. Variance is one of the components of test error, the other is bias. **6. Naive Bayes Classifier:**You can use the Naive Bayes classifier when the features are conditionally independent. I have used it for really simplistic object recognition in RGB where the three channels were assumed to be uncorrelated. **7. Bayesian Nets:**Use Bayesian Nets when you need to model cause-effect relations e.g **medical diagnosis.  
8. K-nearest neighbor:** KNN depends on two things: A metric used to compute the distance between two points and the value of "k" the number of neighbors to consider.

KNN can be used for regression, just average the value for the k nearest neighbors or a point to predict the value for a new point.

One nice advantage of KNN is that it can work fine if you only have a few samples for some of the classes.

**Unsupervised:  
1. Hidden Markov Models:**If your data changes over time, like activity recognition for example; you could use HMM. Similar to ANN, HMM requires you to empirically arrive at a topology.  
I don't know of an alternative to HMM, so maybe you should look it up. **2. Clustering:**Use k-means clustering when you need to group points based on some feature. Use spectral clustering when you need to group points based on connectivity.  
You should also check out semi-supervised learning methods.

**Markov Decision Process:**

A stochastic process containing random variables, transitioning from one state to another depending on certain assumptions and definite probabilistic rules.

These random variables transition from one to state to the other, based on an important mathematical property called Markov Property.

**What Is The Markov Property?**

Discrete Time Markov Property states that the calculated probability of a random process transitioning to the next possible state is only dependent on the current state and time and it is independent of the series of states that preceded it.

The fact that the next possible action/ state of a random process does not depend on the sequence of prior states, renders Markov chains as a memory-less process that solely depends on the current state/action of a variable.

we need to specify two initial measures:

1. An initial probability distribution ( i.e. the start state at time=0, (‘Start’ key))
2. A transition probability of jumping from one state to another (in this case, the probability of transitioning from one token to the other)

<https://www.quora.com/What-actually-is-a-Markov-Process>

-Chandrashekhar

Hadoop Trainer