**Machine Learning ALGO**

**Clustering**

**K\_Means**

Desc -Assumes that the variance to be equal in all directions

Pros - General algo for all type of clustering problems

Cons - Have to input no. Of clusters

- Fail in case the data has blobs present not in clusters(sphreical clusters)

- its not density based technique and takes all the points to be part of n clusters(which we specify) thus it does not take into account the outliers and treats it like the normal data

- Not suitable for Outlier detection

**DBScan**

Desc - Based on density of points present,forms clusters based on density of points.

Pros - Since its tracks down density on its own and makes clusters accordingly so it does not require no of clusters as input

- Good at segregating Outliers as it doesn't require every point to be assigned to a cluster unlike KMeans

Cons - we have to assign 2 params( min\_points -min no of points to conisder a clustering unit[which outgrows]) & (eps - the radius of circle within which to consider the points condition)

- Both these params makes this technique very dependent on them and chaging then can produce varying results

- Moreover eps & min\_points in a way defines the density thereby assuming that the density is constt which is not the case, In cases where the density in multple blobs is varying to a higher degree this will fail

**Classification**

**SVM**

**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.

- The best thing about SVM is that it does not make any strong assumptions on data.

- Robust to noise (because they maximize margins)

- To be used when we have a large no of features & very few obs coz its only concerned with the support vectors

**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.

**SVM (Good to Know Things)**

• If n (features) is large vs. m (training set)

• e.g. text classification problem

Feature vector dimension is 10 000

– Training set is 10 - 1000

– Then use logistic regression or SVM with a linear kernel

If n is small and m is intermediate

– n = 1 - 1000

– m = 10 - 10 000

Gaussian kernel is good

If n is small and m is large

– n = 1 - 1000

– m = 50 000+

SVM will be slow to run with Gaussian kernel

In that case

– Manually create or add more features

– Use logistic regression of SVM with a linear kernel

Logistic regression and SVM with a linear kernel are pretty similar

SVM has a convex optimization problem - so you get a global minimum

**•Weakness of SVM**

It is sensitive to noise

- A relatively small number of mislabeled examples can dramatically decrease the performance

It only considers two classes

- how to do multi-class classificationwith SVM?

- Answer:

1) with output Classes m, learn m SVM’s

 SVM 1 learns “Output==1” vs “Output != 1”

 SVM 2 learns “Output==2” vs “Output != 2”

 :

 SVM m learns “Output==m” vs “Output != m”

(This strategy for prediction of multi-class problems using binary classifiers is known as One-against-all)

2) To predict the output for a new input, just predict with each SVM and find out which one puts

the prediction the furthest into the positive region.

**SOME ISSUES**

Choice of kernel

- Gaussian or polynomial kernel is default

- if ineffective, more elaborated kernels are needed

- domain experts can give assistance in formulating appropriate similarity measures



Choice of kernel parameters

- e.g. σ in Gaussian kernel

- σ is the distance between closest points with different classifications

- In the absence of reliable criteria, applications rely on the use of a validation set or

cross-validation to set such parameters.



Optimization criterion – Hard margin v.s. Soft margin

- a lengthy series of experiments in which various parameters are tested

**Naive Bayes**

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: Preferred for Data with too many categorical var

Weaknesses: Due to their sheer simplicity, NB models are often beaten by models properly trained and tuned using the previous algorithms listed.

**Pros:**

• It is easy and fast to predict class of test data set. It also perform well in multi class prediction

• When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.

• It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

**Cons:**

• If categorical variable has a category (in test data set), which was not observed in training data set,

then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing

techniques is called Laplace estimation.

• On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.

• Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

\* We have to select kernels in NB as well from

- Gaussian - Bernoulli - Multinomial

**K-Nearest Neighbors**

**Strengths**

- Naturally handles multiclass classification and regression

**Weaknesses**

- Performs poorly on high-dimensionality datasets

**Linear Regression**

**Strengths**

- Very fast (runs in constant time)  
- Easy to understand the model  
- Less prone to overfitting

**Pros:**

• K-nearest neighbor algorithm is simple to understand and easy to implement.

• With zero to little training time

• KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting.

• The non-parametric nature of KNN gives it an edge in certain settings where the data may be highly “unusual”.

**Cons:**

• KNN algorithm is the computationally expensive testing phase which is impractical in industry settings. Note the

rigid dichotomy between KNN and the more sophisticated Neural Network which has a lengthy training phase albeit

a very fast testing phase.

**Improvements**

There are many ways in which the KNN algorithm can be improved.

• A simple and effective way to remedy skewed class distributions is by implementing weighed voting. The class of

each of the K neighbors is multiplied by a weight proportional to the inverse of the distance from that point to the

given test point. This ensures that nearer neighbors contribute more to the final vote than the more distant ones.

• Changing the distance metric for different applications may help improve the accuracy of the algorithm. (i.e.

Hamming distance for text classification)

• Rescaling your data makes the distance metric more meaningful. For instance, given 2 features height and weight,

an observation such as x=[180,70]x=[180,70] will clearly skew the distance metric in favor of height. One way of

fixing this is by column-wise subtracting the mean and dividing by the standard deviation. Scikit-

learn’s normalize() method can come in handy.

• Dimensionality reduction techniques like PCA should be executed prior to applying KNN and help make the distance

metric more meaningful.

• Approximate Nearest Neighbor techniques such as using k-d trees to store the training observations can be

leveraged to decrease testing time. Note however that these methods tend to perform poorly in high dimensions

(20+). Try using locality sensitive hashing (LHS) for higher dimensions.

• Furthermore, KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the

training set, it will tend to dominate the majority voting of the new example (large number = more common).

• Finally, the accuracy of KNN can be severely degraded with high-dimension data because there is little difference

between the nearest and farthest neighbor.

**Weakness**

-- Unable to model complex relationships  
-Unable to capture nonlinear relationships without first transforming the inputs

#### Logistic Regression

The models themselves are still "linear," so they work well when your classes are [linearly separable](http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html" \t "/home/rohan/Documents\\x/_blank) (i.e. they can be separated by a single decision surface). Logistic regression can also be regularized by penalizing coefficients with a tunable 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.

**Ensembles**

· **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.

**Random Forests**

Comparable in accuracy to Adaboost, but more robust to errors and outliers

• Insensitive to the number of attributes selected for consideration at

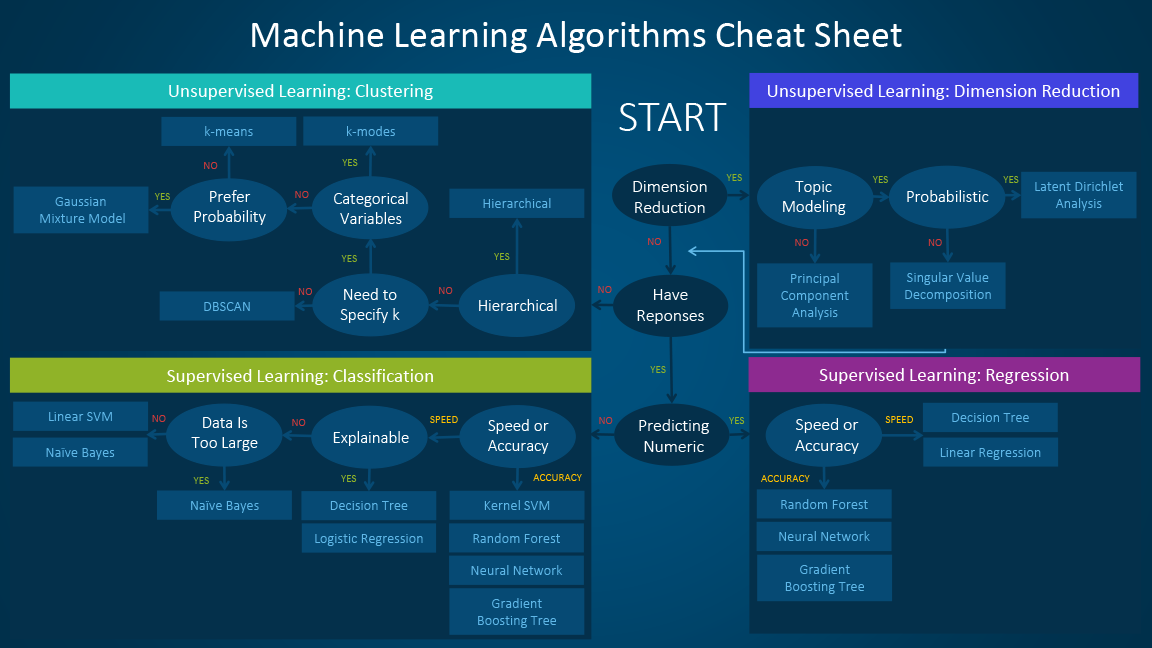
each split, and faster than bagging or boosting• It is a very efficient statistical learning method

• It builds on the idea of bagging, but it provides an improvement

because it de-correlates the trees

**Boosting**

Comparing with bagging: Boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

+++++

### ****Some pros and cons of KNN****

**Pros**:

* No assumptions about data — useful, for example, for nonlinear data
* Simple algorithm — to explain and understand/interpret
* High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models
* Versatile — useful for classification or regression

**Cons**:

* Computationally expensive — because the algorithm stores all of the training data
* High memory requirement
* Stores all (or almost all) of the training data
* Prediction stage might be slow (with big N)
* Sensitive to irrelevant features and the scale of the data

## Pros and Cons associated with SVM

* **Pros:**
  + It works really well with clear margin of separation
  + It is effective in high dimensional spaces.
  + It is effective in cases where number of dimensions is greater than the number of samples.
  + It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
* **Cons:**
  + It doesn’t perform well, when we have large data set because the required training time is higher
  + It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
  + SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

### Advantages and Disadvantages Random Forest:

**Advantages**

Like I already mentioned, an advantage of random forest is that it can be used for both regression and classification tasks and that it’s easy to view the relative importance it assigns to the input features.

Random Forest is also considered as a very handy and easy to use algorithm, because it’s default hyperparameters often produce a good prediction result. The number of hyperparameters is also not that high and they are straightforward to understand.

The Random Forest algorithm can be used for identifying the most important features from the training dataset, in other words, feature engineering.

**Disadvantages**

One of the big problems in machine learning is overfitting, but most of the time this won’t happen that easy to a random forest classifier. That’s because if there are enough trees in the forest, the classifier won’t overfit the model.

The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications the random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred.

And of course Random Forest is a predictive modeling tool and not a descriptive tool. That means, if you are looking for a description of the relationships in your data, other approaches would be preferred.

**ADABOOST**

**https://www.datacamp.com/community/tutorials/adaboost-classifier-python**

## Pros

AdaBoost is easy to implement. It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners. You can use many base classifiers with AdaBoost. AdaBoost is not prone to overfitting. This can be found out via experiment results, but there is no concrete reason available.

## Cons

AdaBoost is sensitive to noise data. It is highly affected by outliers because it tries to fit each point perfectly. AdaBoost is slower compared to XGBoost.

**GBM**

**https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80**

**Random Forests** train each tree independently, using a random sample of the data. This randomness helps to make the model more robust than a single decision tree, and less likely to overfit on the training data. There are typically two parameters in RF - number of trees and no. of features to be selected at each node.  
  
**GBTs** build trees one at a time, where each new tree helps to correct errors made by previously trained tree. With each tree added, the model becomes even more expressive. There are typically three parameters - number of trees, depth of trees and learning rate, and the each tree built is generally shallow.

**Advantages of Naive Bayes:** Super simple, you’re just doing a bunch of counts. If the NB conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. And even if the NB assumption doesn’t hold, a NB classifier still often does a great job in practice. A good bet if want something fast and easy that performs pretty well. Its main disadvantage is that it can’t learn interactions between features (e.g., it can’t learn that although you love movies with Brad Pitt and Tom Cruise, you hate movies where they’re together).

* Very simple, easy to implement and fast.
* If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression.
* Even if the NB assumption doesn’t hold, it works great in practice.
* Need less training data.
* Highly scalable. It scales linearly with the number of predictors and data points.
* Can be used for both binary and mult-iclass classification problems.
* Can make probabilistic predictions.
* Handles continuous and discrete data.
* Not sensitive to irrelevant features.

**Logistic Regression**

t last, here are some points about Logistic regression to ponder upon:

* Does NOT assume a linear relationship between the dependent variable and the independent variables, but it does assume linear relationship between the **logit of the explanatory variables** and the **response**.
* Independent variables can be even the power terms or some other nonlinear transformations of the original independent variables.
* The dependent variable does NOT need to be normally distributed, but it typically assumes a distribution from an exponential family (e.g. binomial, Poisson, multinomial, normal,…); binary logistic regression assume binomial distribution of the response.
* The homogeneity of variance does NOT need to be satisfied.
* Errors need to be independent but NOT normally distributed.
* It uses maximum likelihood estimation (MLE) rather than ordinary least squares (OLS) to estimate the parameters, and thus relies on **large-sample approximations**.

## Advantages / Disadvantages

It is a widely used technique because it is very efficient, does not require too many computational resources, it’s highly interpretable, it doesn’t require input features to be scaled, it doesn’t require any tuning, it’s easy to regularize, and it outputs well-calibrated predicted probabilities.

Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic and also Linear Regression. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train. I typically start with a Logistic Regression model as a benchmark and try using more complex algorithms from there on.

Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is also a good baseline that you can use to measure the performance of other more complex Algorithms.

A disadvantage of it is that we can’t solve non-linear problems with logistic regression since it’s decision surface is linear. Just take a look at the example below that has 2 binary features from 2 examples.