1. Basics
   1. Supervised,unsupervised,reinforcement
   2. Bias-variance trade-off
   3. Overfitting, underfitting
2. Gradient descent:-batch,stochastic
3. Resampling methods
   1. Bootstrapping
   2. Cross-Validation
4. Linear discriminant analysis (LDA)
5. Principal Component Analysis(PCA)
6. Learning Vector Quantization (LVQ)
7. Regularization methods:- Ridge,LASSO
8. Kernel smoothing methods
9. Ensemble learning:-Bagging(bootstrap aggregation),boosting,stacking,blending
10. Ordinary least squares
11. Partial Least squares
12. Kernel density Estimation
13. Radial basis functions
14. Multi co-linearity :Detection(Tolerance,VIF,correlation matrix) and treatment
15. AIC,BIC
16. K-fold cross validation
17. Generalized Additive Models (GAMs)
18. Multivariate Adaptive Regression Splines(MARS)
19. Gradient boosting
20. NLP
    1. Word sense disambiguation
    2. Pronoun resolution
    3. Machine translation
    4. Tokenization
    5. Regular expressions
    6. Stemming
    7. Lemmatization
21. Evaluation metrics
    1. AUC
    2. Precision
    3. Recall
    4. Specificity
    5. Mean absolute percentage error
    6. Root mean square error
22. Algorithms
    1. Linear regression: Usually performed through OLS
    2. Logistic regression
    3. Naive Bayes
    4. K-Nearest Neighbors
    5. K means clustering
    6. Classification and regression trees(CARTs)
    7. Support vector machines
    8. AdaBoost
    9. Random forest
    10. ARIMA
    11. Decision Trees
    12. ID3
    13. CHAID
    14. C4.5, C5.0
    15. Hierarchical Clustering
23. Miscellaneous
    1. Curse of dimensionality
    2. No free lunch theorem
    3. Occams Razor
24. Deep Learning
    1. Neural Networks
    2. Bayesian neural nets
    3. Deep Boltzmann Machine(DBM)
    4. Deep Belief Networks(DBN)
    5. Convolutional Neural Networks
25. Stats
    1. Hypothesis testing
       1. Confidence intervals
       2. P-value
       3. different types of tests
    2. Types of Sampling
    3. Scedasticity

**2) Gradient descent**

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost)

Batch Gradient Descent

One iteration of the algorithm is called one batch and this form of gradient descent is referred to as batch gradient descent. Batch gradient descent is the most common form of gradient descent described in machine learning

Stochastic gradient descent

In this the gradient descent variation the update to the coefficients is performed for each training instance, rather than at the end of the batch of instances

**3 a)Bootstrapping**

In the context of machine learning with bootstrapping, we’re drawing random samples from another sample to generate a new sample that has a balance between the number of samples per class. This is useful when we’d like to model against a dataset with highly unbalanced classes.

**3 b) Cross Validation**

Cross-validation (also called rotation estimation) is a method to estimate how well a model generalizes on a training dataset. In cross-validation we split the training dataset into N number of splits and then separate the splits into training and test groups. We train on the training group of splits and then test the model on the test group of splits. We rotate the splits between the two groups many times until we’ve exhausted all the variations

**4 ) LDA**

Parametric, linear, Classification algorithm used to classify more than two categories

If you have more than two classes, then the Linear Discriminant Analysis is the preferred linear classification technique.

**22 a) Linear Regression**

It is a linear, parametric, regression algorithm.

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares.

OLS:

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients. The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals.

Gradient Descent:

When there are one or more inputs you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data. This operation is called Gradient Descent

Ridge and LASSO:

There are extensions of the training of the linear model called regularization methods. These seek to both minimize the sum of the squared error of the model on the training data (using Ordinary Least Squares) but also to reduce the complexity of the model (like the number or absolute size of the sum of all coefficients in the model).

Two popular examples of regularization procedures for linear regression are:

Lasso Regression: where Ordinary Least Squares is modified to also minimize the absolute sum of the coefficients (called L1 regularization). ˆ

Ridge Regression: where Ordinary Least Squares is modified to also minimize the squared absolute sum of the coefficients (called L2 regularization)

Assumptions for linear regression:

* Linear Assumption Linear regression assumes that the relationship between your input and output is linear.
* Remove Noise
* Remove Collinearity
* Gaussian Distributions. Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution.
* Rescale Inputs: Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization

**22 b) Logistic regression**

It is a linear, parametric classification algorithm. One vs Rest logistic regression can be used to classify multiple categories

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function

Logistic regression models the probability of the default class

Logistic regression is a linear method, but the predictions are transformed using the logistic function

**22 c) Naive bayes**

Classification algorithm

It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) × P(d2|h) and so on. This is a very strong assumption that is most unlikely in real data

**22 d )KNN**

k-Nearest Neighbors (KNN) algorithm is a non-parametric for classification and regression

KNN is used for regression problems the prediction is based on the mean or the median of the K-most similar instances.

**22 f) CART**

It is a nonlinear algorithm

Classically, this algorithm is referred to as decision trees

For classification the Gini cost function is used which provides an indication of how pure the leaf nodes are (how mixed the training data assigned to each node is).

The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node.

You can use pruning after learning your tree to further lift performance. The complexity of a decision tree is defined as the number of splits in the tree. Simpler trees are preferred. They are easy to understand

**22 g) SVM**

Non parametric

support vector machine is a generalization of a simple and intuitive classifier called the maximal margin classifier

support vector machine, which is a further extension of the support vector classifier in order to accommodate non-linear class boundaries

A hyperplane is a subspace whose dimension is one less than that of its ambient space

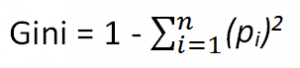
maximal margin hyperplane (also known as the maximal margin hyperplane optimal separating hyperplane), which is the separating hyperplane that optimal separating hyperplane is farthest from the training observations. That is, we can compute the (perpendicular) distance from each training observation to a given separating hyperplane; the smallest such distance is the minimal distance from the observations to the hyperplane, and is known as the margin.

**22 k) Decision Trees**

It is a Non-parametric, nonlinear algorithm

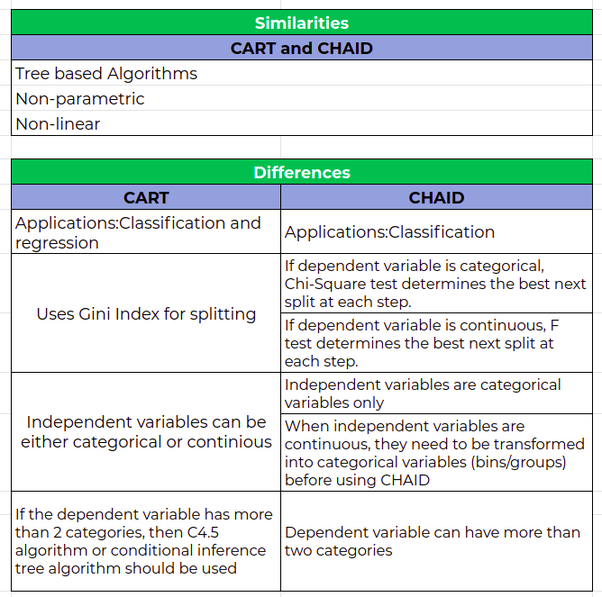
There are many good ways to decide the variable which should be used for splitting. Below are a few:

* Gini Index (used for the popular CART algorithm)
  + A perfect separation results in a Gini score of 0
  + **Formula for Gini Index**



where *pi*is the probability of an object being classified to a particular class.

* Chi Square (used for CHAID algorithm)
  + CHAID (**Ch**i-square **A**utomatic **I**nteraction **D**etector) analysis is an algorithm used for discovering relationships between a categorical response variable and other categorical predictor variables.
* Information Gain/Entropy (used for ID3 algorithm)
* Reduction in Variance



**25 a ii)P- Value**

**Q and A**

**1)Is rotation necessary in PCA?**

Yes, the rotation is necessary because it maximizes the differences between the variance captured by the components

**2)What happens if the components are not rotated in PCA?**

It is a straight effect. If the components are not rotated, then it will diminish eventually, and one must use a lot of various components to explain the data set variance.

**3)Explain why Naive Bayes is so Naive?**

It assumes that all the features in the data set are important, equal and independent.

**4)What is the difference between stochastic gradient descent (SGD) and gradient descent (GD)?**

Both algorithms are methods for finding a set of parameters that minimize a loss function by evaluating parameters against data and then adjusting.

In standard gradient descent, you'll evaluate all training samples for each set of parameters. This is akin to taking big, slow steps toward the solution.

In stochastic gradient descent, you'll evaluate only 1 training sample for the set of parameters before updating them. This is akin to taking small, quick steps toward the solution.

**5)What is the Box-Cox transformation used for?**

The Box-Cox transformation is a generalized "power transformation" that transforms data to make the distribution more normal.

**6)What are 3 ways of reducing dimensionality?**

1. Removing collinear features.

2. Performing PCA, ICA, or other forms of algorithmic dimensionality reduction.

3. Combining features with feature engineering.

**7)How can you choose a classifier based on training set size?**

If the training set is small, high bias / low variance models (E.g. Naive Bayes) tend to perform better because they are less likely to be overfit.

If training set is large, low bias / high variance models (E.g. Logistic Regression) tend to perform better because they can reflect more complex relationships

**8)Explain Principle Component Analysis (PCA).**

PCA is a method for transforming features in a dataset by combining them into uncorrelated linear combinations.

These new features, or principal components, sequentially maximize the variance represented (i.e. the first principal component has the most variance, the second principal component has the second most, and so on).

As a result, PCA is useful for dimensionality reduction because you can set an arbitrary variance cutoff.

**9)What is the advantage of performing dimensionality reduction before fitting an SVM?**

Support Vector Machine Learning Algorithm performs better in the reduced space. It is beneficial to perform dimensionality reduction before fitting an SVM if the number of features is large when compared to the number of observations.

**10)How will you find the correlation between a categorical variable and a continuous variable ?**

You can use the analysis of covariance technqiue to find the correlation between a categorical variable and a continuous variable.

**11)Difference between factor analysis and PCA?**

**12)Explain p-value. Present it as if talking to a client.?**

**13)Explain what a local optimum is and why it is important in a specific context, such as K-means clustering. What are specific ways of determining if you have a local optimum problem? What can be done to avoid local optima?**

**14)What are feature vectors?**

**15)What is: collaborative filtering, n-grams, cosine distance?**

**16)Explain the difference between Gaussian Mixture Model and KMeans.?**

**17)How would you deal with an imbalanced dataset?**

**18)How do you deal with sparse data?**

**19)Explain bootstrapping as if you’re talking to a non-technical person.**

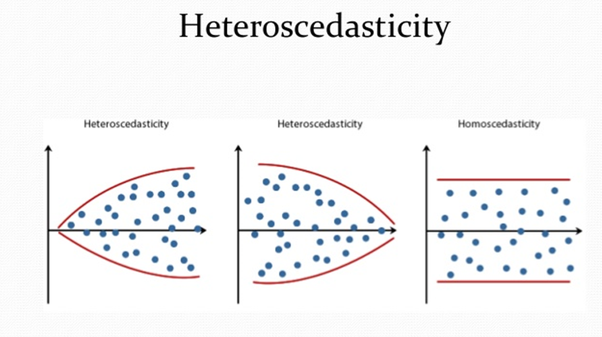
**20)What do you understand by statistical power and how do you calculate it?**

**21)What’s the Central Limit Theorem and what are its practical implications?**

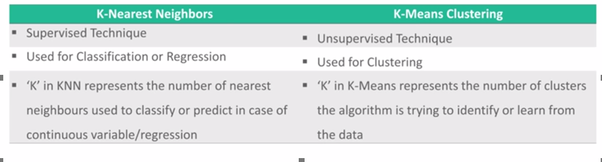
**22)Tell me what heteroscedasticity is and how to solve it?**

If the variability of true values along the regression line is not constant, then this condition is known as heteroskedasticity.

Heteroscedasticity means unequal scatter.



**23)Difference between Kmeans and KNN?**



Good resources:

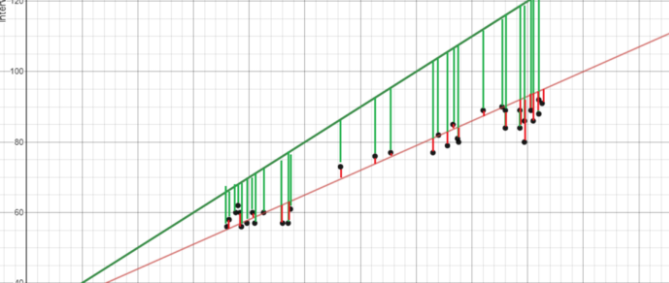
1. <https://remicnrd.github.io./the-machine-learning-cheatsheet/>

***Various Alogs with basic intro:***

**Linear Regression**

Linear Regression is one of the most fundamental algorithms used to model relationships between a dependent variable and one or more independent variables. In simpler terms, it involves finding the ‘line of best fit’ that represents two or more variables.

The line of best fit is found by minimizing the squared distances between the points and the line of best fit — this is known as minimizing the sum of squared residuals. A residual is simply equal to the predicted value minus the actual value.



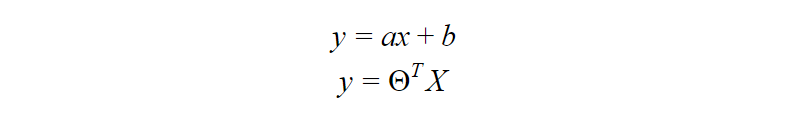
In case it doesn’t make sense yet, consider the image above. Comparing the green line of best fit to the red line, notice how the vertical lines (the residuals) are much bigger for the green line than the red line. This makes sense because the green line is so far away from the points that it isn’t a good representation of the data at all!

If you want to learn more about the math behind linear regression, I would start off with [Brilliant’s explanation](https://brilliant.org/wiki/linear-regression/#:~:text=Linear%20regression%20is%20a%20technique,a%20linear%20relationship%20between%20them.&text=The%20best%2Dfitting%20linear%20relationship%20between%20the%20variables%20x%20and%20y.).

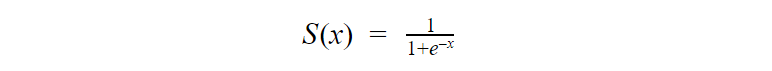
**Logistic Regression**

Logistic regression is similar to linear regression but is used to model the probability of a discrete number of outcomes, typically two. At a glance, logistic regression sounds much more complicated than linear regression, but really only has one extra step.

First, you calculate a score using an equation similar to the equation for the line of best fit for linear regression.

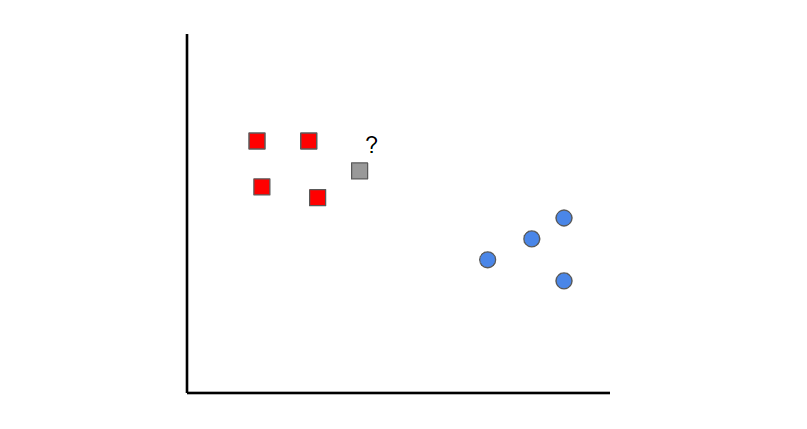


The extra step is feeding the score that you previously calculated in the sigmoid function below so that you get a probability in return. This probability can then be converted to a binary output, either 1 or 0.



To find the weights of the initial equation to calculate the score, methods like gradient descent or maximum likelihood are used. Since it’s beyond the scope of this article, I won’t go into much more detail, but now you know how it works!

**K-Nearest Neighbors**



K-nearest neighbors is a simple idea. First, you start off with data that is already classified (i.e. the red and blue data points). Then when you add a new data point, you classify it by looking at the k nearest classified points. Whichever class gets the most votes determines what the new point gets classified as.

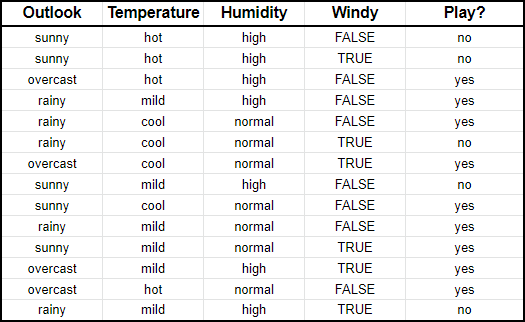
In this case, if we set k=1, we can see that the first nearest point to the grey sample is a red data point. Therefore, the point would be classified as red.

Something to keep in mind is that if the value of k is set too low, it can be subject to outliers. On the other hand, if the value of k is set too high then it might overlook classes with only a few samples.

**Naive Bayes**

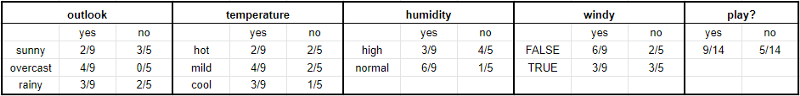
Naive Bayes is a classification algorithm. This means that Naive Bayes is used when the output variable is discrete.

Naive Bayes can seem like a daunting algorithm because it requires preliminary mathematical knowledge in conditional probability and Bayes Theorem, but it’s an extremely simple and ‘naive’ concept, which I’ll do my best to explain with an example:



Suppose we have input data on the characteristics of the weather (outlook, temperature, humidity, windy) and whether you played golf or not (i.e. last column).

What Naive Bayes essentially does is compare the **proportion** between each input variable and the categories in the output variable. This can be shown in the table below.



To give an example to help you read this, in the **temperature** section, it was hot for two days out of the nine days that you played golf (i.e. yes).

In mathematical terms, you can write this as **the probability of it being hot GIVEN that you played golf.**The mathematical notation is P(hot|yes). This is known as conditional probability and is essential to understand the rest of what I’m about to say.

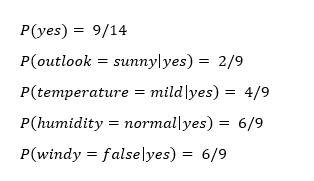
Once you have this, then you can predict whether you’ll play golf or not for any combination of weather characteristics.

Imagine that we have a new day with the following characteristics:

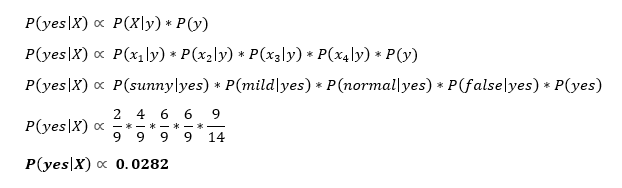
* Outlook: sunny
* Temperature: mild
* Humidity: normal
* Windy: false

First, we’ll calculate the probability that you will play golf given X, P(yes|X) followed by the probability that you won’t play golf given X, P(no|X).

Using the chart above, we can get the following information:



Now we can simply input this information into the following formula:



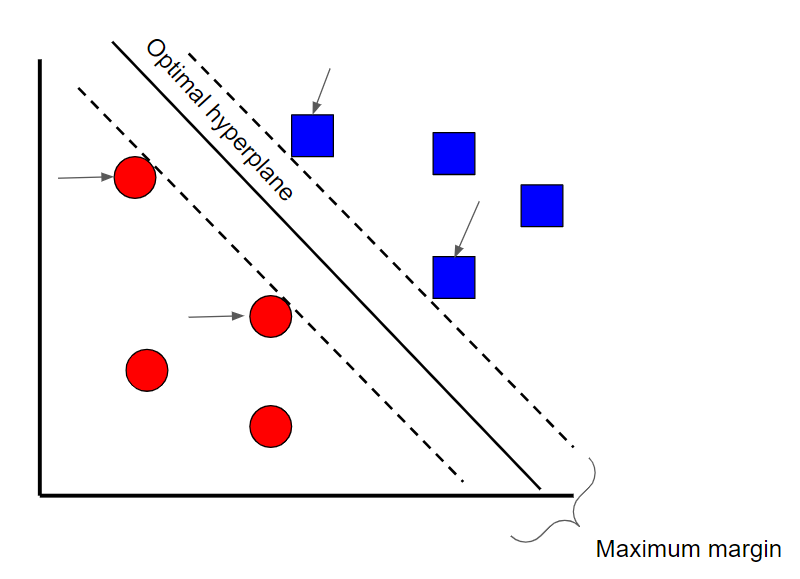
Similarly, you would complete the same sequence of steps for P(no|X).

Image for post

Since P(yes|X) > P(no|X), then you can predict that this person would play golf given that the outlook is sunny, the temperature is mild, the humidity is normal and it’s not windy.

This is the essence of Naive Bayes!

**Support Vector Machines**

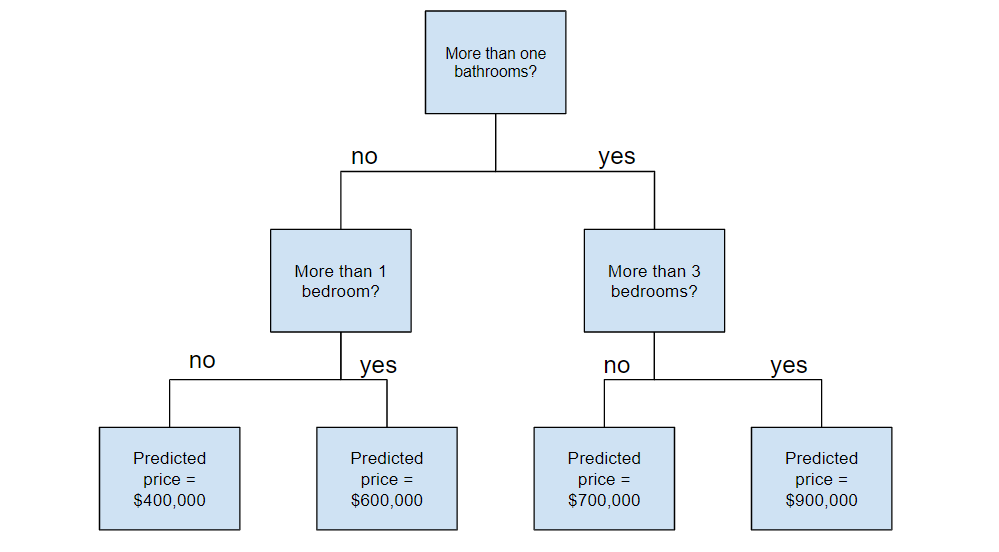


ASupport Vector Machine is a supervised classification technique that can actually get pretty complicated but is pretty intuitive at the most fundamental level. For the sake of this article, we’ll keep it pretty high level.

Let’s assume that there are two classes of data. A support vector machine will find a**hyperplane**or a boundary between the two classes of data that maximizes the margin between the two classes (see above). There are many planes that can separate the two classes, but only one plane can maximize the margin or distance between the classes.

If you want to get into the math behind support vector machines, check out this [series of articles](https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/).

**Decision Tree**



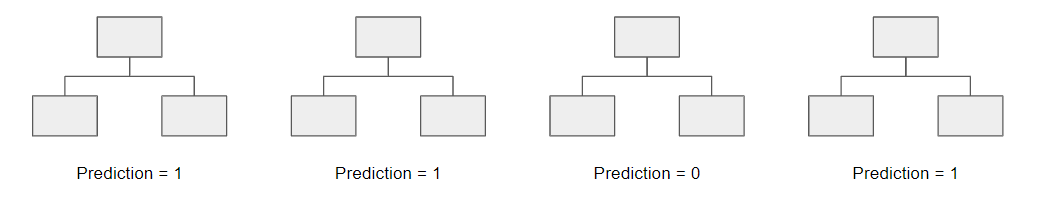
**Random Forest**

Before understanding random forests, there are a couple of terms that you’ll need to know:

* **Ensemble learning** is a method where multiple learning algorithms are used in conjunction. The purpose of doing so is that it allows you to achieve higher predictive performance than if you were to use an individual algorithm by itself.
* **Bootstrap sampling** is a resampling method that uses random sampling with replacement. It sounds complicated but trust me when I say it’s REALLY simple — [read more about it here](https://towardsdatascience.com/what-is-bootstrap-sampling-in-machine-learning-and-why-is-it-important-a5bb90cbd89a).
* **Bagging** when you use the aggregate of the bootstrapped datasets to make a decision — I dedicated an article to this topic so feel free to check it out [here](https://towardsdatascience.com/ensemble-learning-bagging-and-boosting-explained-in-3-minutes-2e6d2240ae21) if this doesn’t make complete sense.

Now that you understand these terms, let’s dive into it.

Random forests are an ensemble learning technique that builds off of decision trees. Random forests involve creating multiple decision trees using bootstrapped datasets of the original data and randomly selecting a subset of variables at each step of the decision tree. The model then selects the mode of all of the predictions of each decision tree (bagging). What’s the point of this? By relying on a “majority wins” model, it reduces the risk of error from an individual tree.

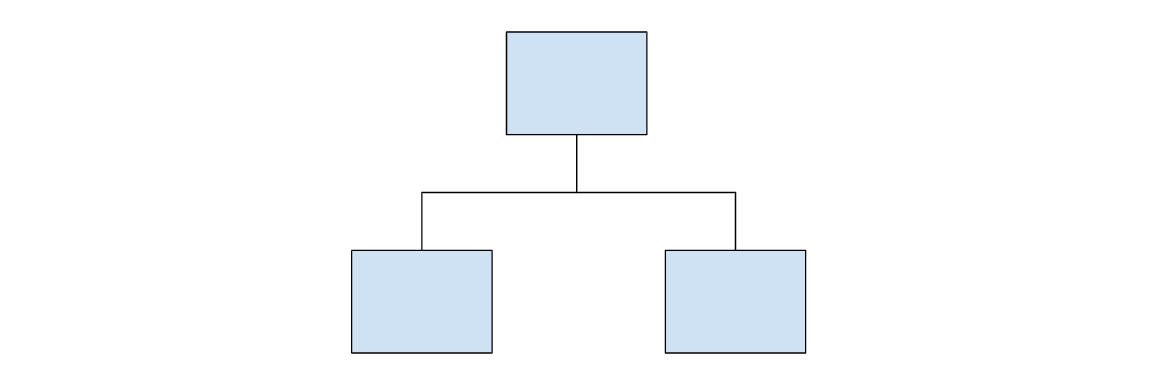


For example, if we created one decision tree, the third one, it would predict 0. But if we relied on the mode of all 4 decision trees, the predicted value would be 1. This is the power of random forests!

**AdaBoost**

AdaBoost, or Adaptive Boost, is also an ensemble algorithm that leverages bagging and boostingmethods to develop an enhanced predictor.

AdaBoost is similar to Random Forests in the sense that the predictions are taken from many decision trees. However, there are three main differences that make AdaBoost unique:



Example of a stump

1. First, AdaBoost creates a forest of stumps rather than trees. A stump is a tree that is made of only one node and two leaves (like the image above).
2. Second, the stumps that are created are not equally weighted in the final decision (final prediction). Stumps that create more error will have less say in the final decision.
3. Lastly, the order in which the stumps are made is important, because each stump aims to reduce the errors that the previous stump(s) made.

In essence, AdaBoost takes a more iterative approach in the sense that it seeks to iteratively improve from the mistakes that the previous stump(s) made.

**Gradient Boost**

It’s no surprise that Gradient **Boost**is also an ensemble algorithm that uses boostingmethods to develop an enhanced predictor. In many ways, Gradient Boost is similar to AdaBoost, but there are a couple of key differences:

* Unlike AdaBoost which builds stumps, Gradient Boost builds trees with usually 8–32 leaves.
* Gradient Boost views the boosting problem as an optimization problem, where it uses a loss function and tries to minimize the error. This is why it’s called **Gradient** boost, as it’s inspired by gradient descent.
* Lastly, the trees are used to predict the residuals of the samples (predicted minus actual).

While the last point may have been confusing, all that you need to know is that Gradient Boost starts by building one tree to try to fit the data, and the subsequent trees built after aim to reduce the residuals (error). It does this by concentrating on the areas where the existing learners performed poorly, similar to AdaBoost.

**XGBoost**

XGBoost is one of the most popular and widely used algorithms today because it is simply so powerful. It is similar to Gradient Boost but has a few extra features that make it that much stronger including…

* **A proportional shrinking of leaf nodes (pruning)**— used to improve the generalization of the model
* **Newton Boosting** — provides a direct route to the minima than gradient descent, making it much faster
* **An extra randomization parameter**— reduces the correlation between trees, ultimately improving the strength of the ensemble
* **Unique penalization of trees**

I strongly recommend that you watch [StatQuest’s video](https://www.youtube.com/watch?v=OtD8wVaFm6E&t=3s) to understand how the algorithm works in greater detail.

**LightGBM**

If you thought XGBoost was the best algorithm out there, think again. LightGBM is another type of boosting algorithm that has shown to be faster and sometimes more accurate than XGBoost.

What makes LightGBM different is that it uses a unique technique called Gradient-based One-Side Sampling (GOSS) to filter out the data instances to find a split value. This is different than XGBoost which uses pre-sorted and histogram-based algorithms to find the best split.

[*Read more about Light GBM vs XGBoost here*](https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db)*!*

**CatBoost**

CatBoost is another algorithm based on Gradient Descent that has a few subtle differences that make it unique:

* CatBoost implements symmetric trees which help in decreasing prediction time and it also has a shallower tree-depth by default (six)
* CatBoost leverages random permutations similar to the way XGBoost has a randomization parameter
* Unlike XGBoost however, CatBoost handles categorical features more elegantly, using concepts like ordered boosting and response coding

Overall, what makes CatBoost so powerful is its low latency requirements which translates to it being around **eight times faster** than XGBoost.

**Radius Neighbors Classifier**

In scikit-learn RadiusNeighborsClassifier is very similar to KNeighborsClassifier with the exception of two parameters. First, in RadiusNeighborsClassifier we need to specify the radius of the fixed area used to determine if an observation is a neighbor using radius.

Unless there is some substantive reason for setting radius to some value, it is best to treat it like any other hyperparameter and tune it during model selection. The second useful parameter is outlier\_label, which indicates what label to give an observation that has no observations within the radius – which itself can often be a useful tool for identifying outliers.

**Passive Aggressive Classifier**

PA algorithm is a margin based online learning algorithm for binary classification. Unlike PA algorithm, which is a hard-margin based method, PA-I algorithm is a soft margin based method and robuster to noise.

**BernoulliNB**

Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features.

**ExtraTreeClassifier**

ExtraTreesClassifier is an ensemble learning method fundamentally based on decision trees. ExtraTreesClassifier, like RandomForest, randomizes certain decisions and subsets of data to minimize over-learning from the data and overfitting. Let’s look at some ensemble methods ordered from high to low variance, ending in ExtraTreesClassifier.

**Bagging classifier**

Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (E.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

**Quadratic Discriminant Analysis**

A classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes’ rule. The model fits a Gaussian density to each class.

**LSTM Neural Network**

LSTM Neural Network is a variant of the Recurrent Neural Network. LSTM units effectively solve the problem of vanishing gradient. They are typically used for sequential classification problems: text labeling, speech recognition. The Encoder-Decoder architecture of LSTM networks allows building machine translation systems.

**Expectation Maximization**

Expectation Maximization, as well as Latent Dirichlet Allocation (below), are both examples of the so-called "soft" clustering. They assign one example to multiple clusters with a probability of membership. Expectation Maximization learns Gaussian Mixture Models which solves the problem of K-Means: the clusters can be non-spherical.

**DBSCAN**

DBSCAN (for *Density-Based Spatial Clustering of Applications with Noise*) is another popular clustering algorithm. Its difference from K-Means lies in the fact that DBSCAN doesn't use the notion of cluster centroids. It's density based, so the number of clusters depends on the data itself. Clusters can have absolutely any form: for DBSCAN a cluster is a subset of points that form a dense "cloud".

**Latent Dirichlet Allocation**

LDA is one the most important unsupervised learning algorithms. It was successfully applied to text analysis (topic modeling), social network analysis (community overlaps), content recommendation, genetic population analysis, and many others.

In the nutshell, LDA takes a collection of documents, the number of topics, and returns a distribution of topics over documents and a distribution of words over topics.

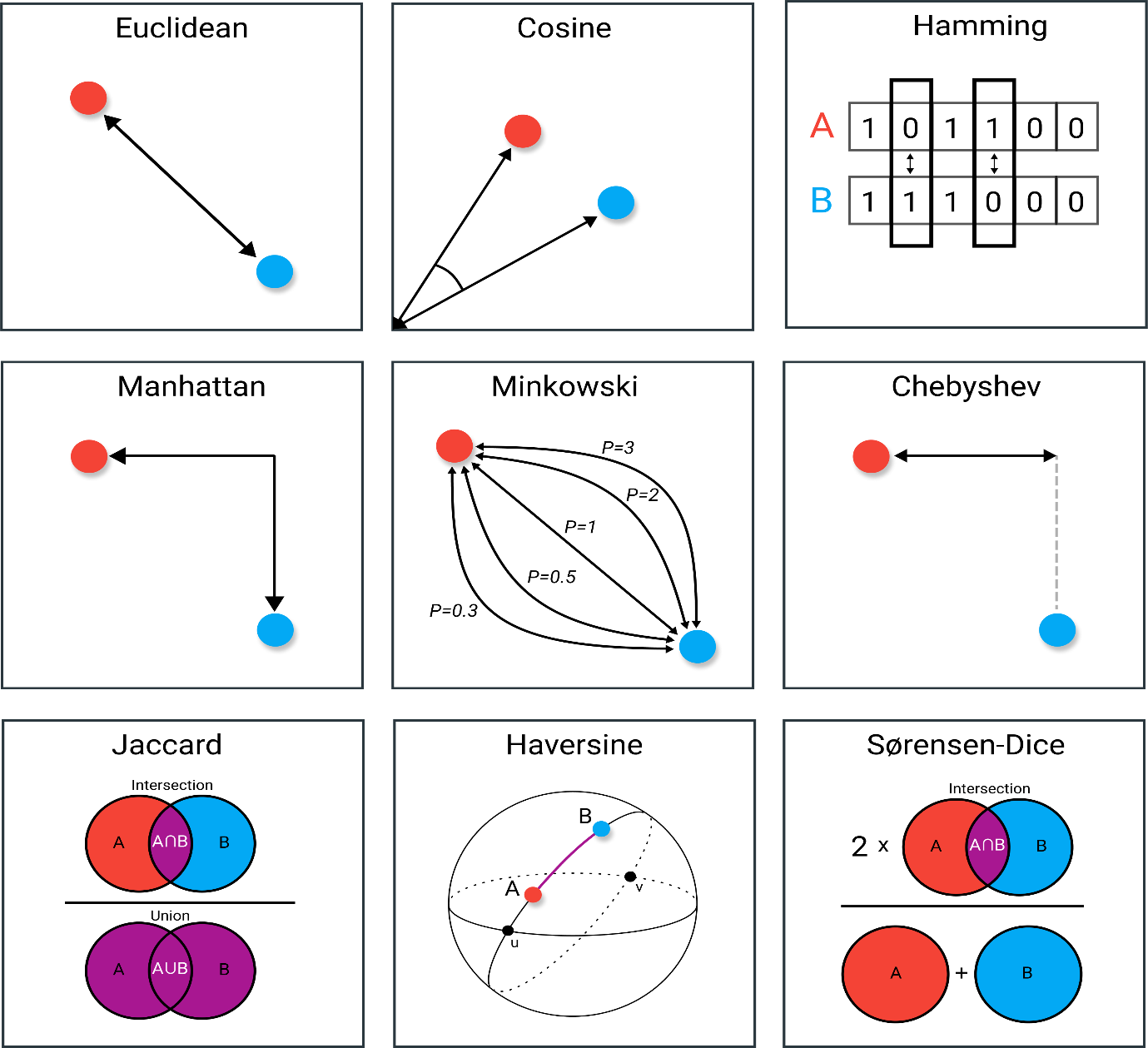
LDA can be used as a soft-clustering method: every topic can be seen as a cluster and each document can have several topics with different probabilities, thus can belong to several clusters.

**UMAP**

One important aspect of the machine learning practice is data visualization and dimensionality reduction. Dimensionality reduction can be done to visualize data (the human can only see up to three-dimensional data) and to make learning tractable and/or more accurate.

For a long [Principal Component Analysis](https://en.wikipedia.org/wiki/Principal_component_analysis) or PCA was the most popular tool for dimensionality reduction and [t-SNE](https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding) was used to visualize data. However, in 2018 the new algorithm, [UMAP](https://github.com/lmcinnes/umap) (for *Uniform Manifold Approximation and Projection*) was proposed and efficiently implemented. UMAP combines the advantage of both PCA and t-SNE: it can be used as a dimensionality reduction techniques (t-SNE can't) and has visualization properties similar or surpassing those of t-SNE on many tasks.

***Satistical Distances :***



Many algorithms, whether supervised or unsupervised, make use of **distance measures**. These measures, such as euclidean distance or cosine similarity, can often be found in algorithms such as k-NN, UMAP, HDBSCAN, etc.

Understanding the field of distance measures is more important than you might realize. Take k-NN for example, a technique often used for supervised learning. As a default, it often uses euclidean distance. By itself, a great distance measure.

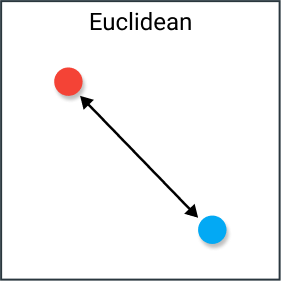
However, what if your data is highly dimensional? Would euclidean distance then still work? Or what if your data consists of geospatial information? Perhaps haversine distance would then be a better alternative!

*Knowing when to use which distance measure can help you go from a poor classifier to an accurate model.*

In this article, we will go through many distance measures and explore how and when they best can be used. Most importantly, I will be talking about their disadvantages so that you can recognize when to steer clear of certain measures.

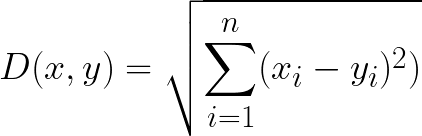
**NOTE**: For most distance measures long elaborate papers could and have been written on their use-cases, advantages, and disadvantages. I will try to cover as much as possible but may fall short! Thus, consider this article a global overview of these measures.

***1. Euclidean Distance***



We start with the most common distance measure, namely Euclidean distance. It is a distance measure that best can be explained as the length of a segment connecting two points.

The formula is rather straightforward as the distance is calculated from the cartesian coordinates of the points using the Pythagorean theorem.



Euclidean distance

**Disadvantages**

Although it is a common distance measure, Euclidean distance is not scale in-variant which means that distances computed might be skewed depending on the units of the features. Typically, one needs to **normalize**the data before using this distance measure.

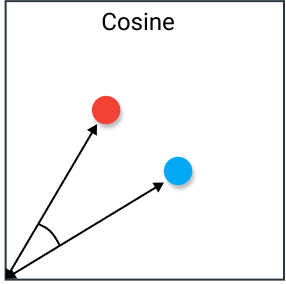
Moreover, as the dimensionality increases of your data, the less useful Euclidean distance becomes. This has to do with the curse of dimensionality which relates to the notion that higher-dimensional space does not act as we would, intuitively, expect from 2- or 3-dimensional space. For a good summary, see [this](https://stats.stackexchange.com/questions/99171/why-is-euclidean-distance-not-a-good-metric-in-high-dimensions) post.

Use Cases

Euclidean distance works great when you have low-dimensional data and the magnitude of the vectors is important to be measured. Methods like kNN and HDBSCAN show great results out of the box if Euclidean distance is used on low-dimensional data.

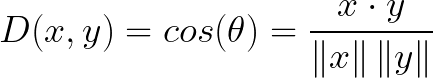
Although many other measures have been developed to account for the disadvantages of Euclidean distance, it is still one of the most used distance measures for good reasons. It is incredibly intuitive to use, simple to implement and shows great results in many use-cases.

***2. Cosine Similarity***



Cosine similarity has often been used as a way to counteract Euclidean distance’s problem with high dimensionality. The cosine similarity is simply the cosine of the angle between two vectors. It also has the same inner product of the vectors if they were normalized to both have length one.

Two vectors with exactly the same orientation have a cosine similarity of 1, whereas two vectors diametrically opposed to each other have a similarity of -1. Note that their magnitude is not of importance as this is a measure of orientation.



Cosine Similarity

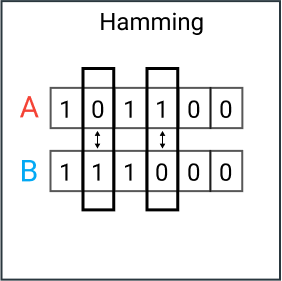
Disadvantages

One main disadvantage of cosine similarity is that the magnitude of vectors is not taken into account, merely their direction. In practice, this means that the differences in values are not fully taken into account. If you take a recommender system, for example, then the cosine similarity does not take into account the difference in rating scale between different users.

Use Cases

We use cosine similarity often when we have high-dimensional data and when the magnitude of the vectors is not of importance. For text analyses, this measure is quite frequently used when the data is represented by word counts. For example, when a word occurs more frequently in one document over another this does not necessarily mean that one document is more related to that word. It could be the case that documents have uneven lengths and the magnitude of the count is of less importance. Then, we can best be using cosine similarity which disregards magnitude.

***3. Hamming Distance***



Hamming distance is the number of values that are different between two vectors. It is typically used to compare two binary strings of equal length. It can also be used for strings to compare how similar they are to each other by calculating the number of characters that are different from each other.

**Disadvantages**

As you might expect, hamming distance is difficult to use when two vectors are not of equal length. You would want to compare same-length vectors with each other in order to understand which positions do not match.

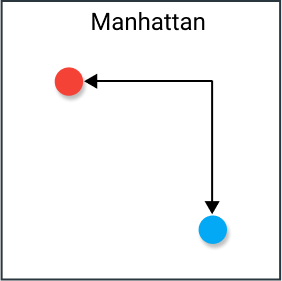
Moreover, it does not take the actual value into account as long as they are different or equal. Therefore, it is not advised to use this distance measure when the magnitude is an important measure.

**Use Cases**

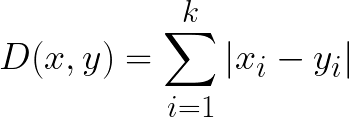
Typical use cases include error correction/detection when data is transmitted over computer networks. It can be used to determine the number of distorted bits in a binary word as a way to estimate error.

Moreover, you can also use Hamming distance to measure the distance between categorical variables.

***4. Manhattan Distance***



The Manhattan distance, often called Taxicab distance or City Block distance, calculates the distance between real-valued vectors. Imagine vectors that describe objects on a uniform grid such as a chessboard. Manhattan distance then refers to the distance between two vectors if they could only move right angles. There is no diagonal movement involved in calculating the distance.



Manhattan distance

**Disadvantages**

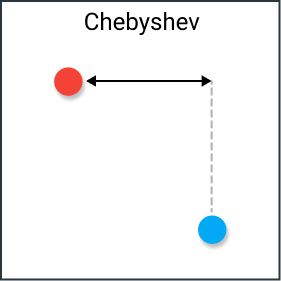
Although Manhattan distance seems to work okay for [high-dimensional data](https://www.quora.com/What-is-the-difference-between-Manhattan-and-Euclidean-distance-measures), it is a measure that is somewhat less intuitive than euclidean distance, especially when using in high-dimensional data.

Moreover, it is more likely to give a higher distance value than euclidean distance since it does not the shortest path possible. This does not necessarily give issues but is something you should take into account.

**Use Cases**

When your dataset has discrete and/or binary attributes, Manhattan seems to work quite well since it takes into account the paths that realistically could be taken within values of those attributes. Take Euclidean distance, for example, would create a straight line between two vectors when in reality this might not actually be possible.

***5. Chebyshev Distance***



Chebyshev distance is defined as the greatest of difference between two vectors along any coordinate dimension. In other words, it is simply the maximum distance along one axis. Due to its nature, it is often referred to as Chessboard distance since the minimum number of moves needed by a king to go from one square to another is equal to Chebyshev distance.



Chebyshev distance

**Disadvantages**

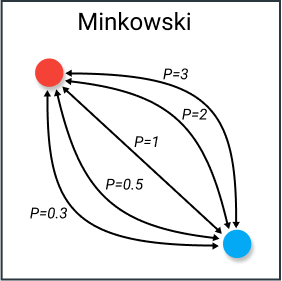
Chebyshev is typically used in very specific use-cases, which makes it difficult to use as an all-purpose distance metric, like Euclidean distance or Cosine similarity. For that reason, it is suggested to only use it when you are absolutely sure it suits your use-case.

**Use Cases**

As mentioned before, Chebyshev distance can be used to extract the minimum number of moves needed by a king to go from one square to another. Moreover, it can be a useful measure in games that allow unrestricted 8-way movement.

In practice, Chebyshev distance is often used in warehouse logistics as it closely resembles the time an overhead crane takes to move an object.

***6. Minkowski***

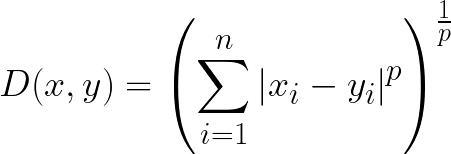


Minkowski distance is a bit more intricate measure than most. It is a metric used in Normed vector space (n-dimensional real space), which means that it can be used in a space where distances can be represented as a vector that has a length.

This measure has three requirements:

* **Zero Vector**— The zero vector has a length of zero whereas every other vector has a positive length. For example, if we travel from one place to another, then that distance is always positive. However, if we travel from one place to itself, then that distance is zero.
* **Scalar Factor**— When you multiple the vector with a positive number its length is changed whilst keeping its direction. For example, if we go a certain distance in one direction and add the same distance, the direction does not change.
* **Triangle Inequality**— The shortest distance between two points is a straight line.

The formula for the Minkowski distance is shown below:



Minkowski distance

Most interestingly about this distance measure is the use of parameter **p**. We can use this parameter to manipulate the distance metrics to closely resemble others.

Common values of p are:

* p=1 — Manhattan distance
* p=2 — Euclidean distance
* p=*∞*— Chebyshev distance

**Disadvantages**

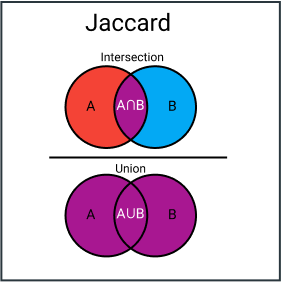
Minkowski has the same disadvantages as the distance measures they represent, so a good understanding of metrics like Manhattan, Euclidean, and Chebyshev distance is extremely important.

Moreover, the parameter p can actually be troublesome to work with as finding the right value can be quite computationally inefficient depending on your use-case.

**Use Cases**

The upside to p is the possibility to iterate over it and find the distance measure that works best for your use case. It allows you a huge amount of flexibility over your distance metric, which can be a huge benefit if you are closely familiar with p and many distance measures.

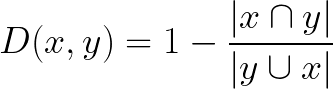
***7. Jaccard Index***



The Jaccard index (or Intersection over Union) is a metric used to calculate the similarity and diversity of sample sets. It is the size of the intersection divided by the size of the union of the sample sets.

In practice, it is the total number of similar entities between sets divided by the total number of entities. For example, if two sets have 1 entity in common and there are 5 different entities in total, then the Jaccard index would be 1/5 = 0.2.

To calculate the Jaccard distance we simply subtract the Jaccard index from 1:



Jaccard distance

**Disadvantages**

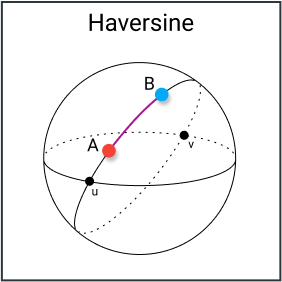
A major disadvantage of the Jaccard index is that it is highly influenced by the size of the data. Large datasets can have a big impact on the index as it could significantly increase the union whilst keeping the intersection similar.

**Use-Cases**

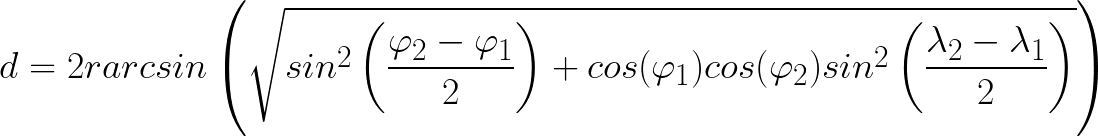
The Jaccard index is often used in applications where binary or binarized data are used. When you have a deep learning model predicting segments of an image, for instance, a car, the Jaccard index can then be used to calculate how accurate that predicted segment given true labels.

Similarly, it can be used in text similarity analysis to measure how much word choice overlap there is between documents. Thus, it can be used to compare sets of patterns.

***8. Haversine***



Haversine distance is the distance between two points on a sphere given their longitudes and latitudes. It is very similar to Euclidean distance in that it calculates the shortest line between two points. The main difference is that no straight line is possible since the assumption here is that the two points are on a sphere.



Haversine distance between two points

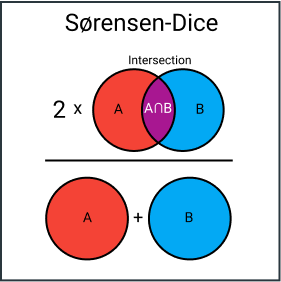
**Disadvantages**

One disadvantage of this distance measure is that it is assumed the points lie on a **sphere**. In practice, this is seldom the case as, for example, the earth is not perfectly round which could make calculation in certain cases difficult. Instead, it would be interesting to look towards **Vincenty distance** which assumes an ellipsoid instead.

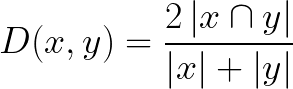
**Use Cases**

As you might have expected, Haversine distance is often used in navigation. For example, you can use it to calculate the distance between two countries when flying between them. Note that it is much less suited if the distances by themselves are already not that large. The curvature will not have that large of an impact.

***9. Sørensen-Dice Index***



The Sørensen-Dice index is very similar to Jaccard index in that it measures the similarity and diversity of sample sets. Although they are calculated similarly the Sørensen-Dice index is a bit more intuitive because it can be seen as the percentage of overlap between two sets, which is a value between 0 and 1:



Sørensen–Dice coefficient

**Disadvantages**

Like the Jaccard index, they both overstate the importance of sets with little to no ground truth positive sets. As a result, it could dominate the average score taken over multiple sets. It weights each item inversely proportionally to the size of the relevant set rather than treating them equally.

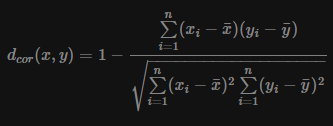
**Use Cases**

The use cases are similar, if not the same, as Jaccard index. You will find it typically used in either image segmentation tasks or text similarity analyses.

**NOTE:**There are many more distance measures than the 9 mentioned here. If you are looking for more interesting metrics, I would suggest you look into one of the following: **Mahalanobis, Canberra, Braycurtis, and KL-divergence.**

Other dissimilarity measures exist such as **correlation-based distances**, which is widely used for gene expression data analyses. Correlation-based distance is defined by subtracting the correlation coefficient from 1. Different types of correlation methods can be used such as:

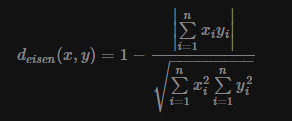
***10. Pearson correlation distance:***



Pearson correlation measures the degree of a linear relationship between two profiles.

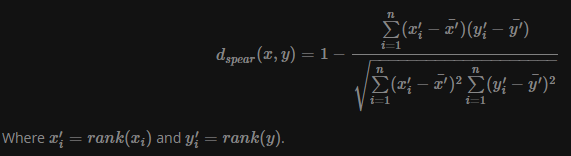
***11. Eisen cosine correlation distance (Eisen et al., 1998):***

It’s a special case of Pearson’s correlation with x̅ and y̅ both replaced by zero:

******

***12. Spearman correlation distance:***

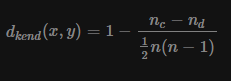
The spearman correlation method computes the correlation between the rank of x and the rank of y variables.



***13. Kendall correlation distance:***

Kendall correlation method measures the correspondence between the ranking of x and y variables. The total number of possible pairings of x with y observations is n(n−1)/2n(n−1)/2, where n is the size of x and y. Begin by ordering the pairs by the x values. If x and y are correlated, then they would have the same relative rank orders. Now, for each yi, count the number of yj > yi (concordant pairs (c)) and the number of yj < yi (discordant pairs (d)).

Kendall correlation distance is defined as follow:



Where,

* nc: total number of concordant pairs
* nd: total number of discordant pairs
* n: size of x and y

Note that,

* Pearson correlation analysis is the most commonly used method. It is also known as a parametric correlation which depends on the distribution of the data.
* Kendall and Spearman correlations are non-parametric and they are used to perform rank-based correlation analysis.

In the formula above, *x* and *y* are two vectors of length *n* and, means x̅ and y̅, respectively. The distance between x and y is denoted *d*(*x*, *y*).

***Guide to which Clustering Distance/ Model to use in the wild:***

[1] Connectivity models: Builds models based on distance connectivity. Eg hierarchical clustering. Used when we need different partitioning based on tree cut height. R function: hclust in stats package.

[2] Centroid models: Builds models by representing each cluster by a single mean vector. Used when we need crisp partitioning (as opposed to fuzzy clustering described later). R function: k - means in stats package.

[3] Distribution models: Builds models based on statistical distributions such as multivariate normal distributions used by the expectation-maximization algorithm. Used when cluster shapes can be arbitrary unlike k-means which assumes circular clusters. R function: e-m cluster in the e-m cluster package.

[4] Density models: Builds models based on clusters as connected dense regions in the data space. E.g. DBSCAN and OPTICS. Used when cluster shapes can be arbitrary unlike k-means which assumes circular clusters. R function db scan in package db scan.

[5] Subspace models: Builds models based on both cluster members and relevant attributes. E.g. biclustering (also known as co-clustering or two-mode-clustering). Used when simultaneous row and column clustering is needed. R function biclust in biclust package.

[6] Group models: Builds models based on the grouping information. E.g. collaborative filtering (recommender algorithm). R function Recommender in recommenderlab package.

[7] Graph-based models: Builds models based on clique. Community structure detection algorithms try to find dense subgraphs in directed or undirected graphs. E.g. R function cluster\_walktrap in igraph package.

[8] Kohonen Self-Organizing Feature Map: Builds models based on neural network. R function som in the kohonen package.

[9] Spectral Clustering: Builds models based on non-convex cluster structure, or when a measure of the center is not a suitable description of the complete cluster. R function specc in the kernlab package.

[10] subspace clustering : For high-dimensional data, distance functions could be problematic. cluster models include the relevant attributes for the cluster. E.g., hddc function in the R package HDclassif.

[11] Sequence clustering: Group sequences that are related. rBlast package.

[12] Affinity propagation: Builds models based on message passing between data points. It does not require the number of clusters to be determined before running the algorithm. It is better for certain computer vision and computational biology tasks, E.g. clustering of pictures of human faces and identifying regulated transcripts, than k-means, [Ref](http://science.sciencemag.org/content/315/5814/972) Rpackage APCluster.

[13] Stream clustering: Builds models based on data that arrive continuously such as telephone records, financial transactions etc. E.g. R package BIRCH [[https://cran.r-project.org/src/contrib/Archive/birch/]](https://cran.r-project.org/src/contrib/Archive/birch/%5d)

[14] Document clustering (or text clustering): Builds models based on SVD. It has used in topic extraction. E.g. Carrot [[http://search.carrot2.org]](http://search.carrot2.org]/) is an open source search results clustering engine which can cluster documents into thematic categories.

[15] Latent class model: It relates a set of observed multivariate variables to a set of latent variables. LCA may be used in collaborative filtering. R function Recommender in recommenderlab package has collaborative filtering functionality.

[16] Biclustering: Used to simultaneously cluster rows and columns of two-mode data. E.g. R function biclust in package biclust.

[17] Soft clustering (fuzzy clustering): Each object belongs to each cluster to a certain degree. E.g. R Fclust function in the fclust package.

