# Analysis and Visualization of The MAGIC Gamma Ray Telescope dataset By using Machine Learning And testing multiple Classification algorithms.

**By** Prateek Tripathi
Registration no.: 11908181
Email: prateek.11908181@lpu.in

School of Computer Science and Engineering,
Lovely Faculty of Technology and Sciences,
Lovely Professional University
Phagwara, Punjab, India

# INDEX

**Abstract:**

The MAGIC telescope is presently at its commissioning phase at the Roque de los Muchachos Observatory (ORM) on the island of La Palma. MAGIC will become the largest ground-based gamma ray telescope in the world, being sensitive to photons of energies as low as 30 GeV. The spectral range between 10 and 300 GeV remains to date mostly unexplored. Observations in this region of the spectrum are expected to provide key data for the understanding of a wide variety of astrophysical phenomena belonging to the so-called "non thermal Universe", like the processes in the nuclei of active galaxies, the radiation mechanisms of pulsars and supernova remnants, and the enigmatic gamma-ray bursts. And overview of the telescope and its Physics goals is presented.

**Key word:** Gamma ray astronomy– Cherenkov telescopes

## 1. Introduction to The MAGIC Telescope:

The MAGIC telescope the 17 m diameter f/1 MAGIC telescope is the largest of the new generation IACTs. MAGIC has been built in the Canarian Island of La Palma (28.8 N, 17.9 W) at the Roque de los Muchachos observatory (ORM), 2200 m above sea level. Its 234 m2 parabolic dish is composed of 956 49.5 × 49.5 cm2 all-aluminium spherical mirror tiles mounted on a lightweight (< 10 ton) carbon fibre frame.

The parabolic shape was chosen to minimize the time spread of the Cherenkov light flashes on the camera plane, which allows to reduce the rate of fake events induced by night-sky background light. Each mirror is made of an aluminium honeycomb structure, on which a 5 mm plate of AlMgSi1.0 alloy is glued. The aluminium plate of each tile is diamond-milled to achieve the spherical reflecting surface with the radius of curvature most adequate for its position on the paraboloid.

A thin quartz layer protects the mirror surface from aging. Mirrors are grouped in panels of three or four, which can be oriented during the telescope operation through an active mirror control system to correct for the possible deformations of the telescope structure. While all other new generation Cherenkov telescopes aim at the improvement of sensitivity and energy resolution in the 100 GeV regime by using stereoscopic systems of relatively small (10 m) telescopes, MAGIC, through the choice of a single huge reflector, will achieve the lowest energy threshold among IACTs, of about 30 GeV. Its altazimuth mount can point to anywhere in the sky in less than 20 seconds, a unique feature which is essential for the study of transient events like GRBs.

### 1.1. The Camera

The MAGIC camera is equipped with 576 6dynode compact photomultipliers (PMs), of 20% average quantum efficiency in the 300500 nm range (Ostankov et al. 2000). Each PM is coupled to a small light collecting cone to maximize the active surface of the camera. The total field of view of the camera is about 4◦, divided into two sections: an inner hexagon of 396 small pixels of about 3 cm (0.1◦) diameter, and four outer rings of 6 cm diameter PMs. The use of larger pixels in the outer zone reduces the cost of the camera, while the quality of the image, already limited in this zone by the coma aberration, is not deteriorated significantly.

### 1.2. Trigger and Readout Electronics

The analog signals from the PMs are transformed into light pulses using VCSELs (Vertical Cavity Surface Emitting Lasers), and transported via optical links to the control house, about 100 m away from the telescope. Optical signal transport reduces the need for heavy coaxial cables and guarantees minimal degradation of the pulse shape. A fast 2-level trigger system (Bastieri et al. 2001) starts the data acquisition (DAQ) whenever N neighbouring pixels fire within a coincidence window Δt of a few nanoseconds. All the trigger parameters will be fine-tuned in the commissioning phase of the telescope. Simulations show that reasonable values could be N = 4, Δt = 6ns for a pixel threshold of about 8 photoelectrons.

A programmable second level trigger, which can sustain rates of up to 1 MHz, offers the possibility of using fast ($\simeq 60 - 80$ ns) pattern recognition routines to perform some background rejection already at trigger level. The pixel signals are stretched, split into a high- and a low-gain branch (to enlarge the dynamic range), and then digitized using fast (300 MHz), 8-bit Flash ADCs. The maximum event readout rate of the DAQ will be about 1 kHz. As of May 2003, the first tests of the whole DAQ chain and other subsystems are under way at the ORM. About 40% of the mirror surface is already installed, the camera and the trigger are fully operational, and the readout of all 576 channels is expected to be working by the Summer. Calibration runs on the Crab Nebula (the standard candle in high energy gamma-ray astronomy) will start in October, and the first observational campaign is due for 2004.

## 2. The Physics goals of MAGIC

The most relevant Physics questions that will be addressed by MAGIC are the following:

### 2.1. The Gamma-Ray Horizon

Due to the absorption of gamma-rays through interaction with the extragalactic background light, only a few nearby blazars have been observed up to now by ground-based gamma telescopes. The low threshold of MAGIC will extend the observable gamma Universe well beyond the limits of the present TeV instruments. Furthermore, if a large sample of AGNs at different redshifts is detected by MAGIC, an indirect measurement of the infrared background density may also be feasible.

### 2.2. Active Galactic Nuclei

The observation of nearby blazars at TeV energies with the previous generation of IACTs have been extremely fruitful. The fast flux variations observed from BL Lacs Mkn 421 and Mkn 501 (Gaidos at al. 1996; Aharonian et al. 1999b) indicate that the emission region is very small, less than one light-day (Hillas 1999), and probably very close to the central supermassive black hole. MAGIC will provide better data on these and other similar objects and will take part in multi-wavelength observation campaigns which will contribute to the understanding of the emission processes occurring in the jets of AGNs. A recent result from the HEGRA collaboration on the giant radio galaxy M87 (Aharonian et al. 2003) hints at the possibility that blazars are not the only subclass of AGNs capable of emitting GeV-TeV radiation.

### 2.3. Gamma Ray Pulsars

Two different models have been proposed to explain the mechanism of the gamma-ray emission observed by EGRET from six galactic pulsars. The models differ on the location of the emission region: near the magnetic poles (polar cap) or in the outer part of the pulsar magnetosphere (outer gap). The two theories predict slightly different cut-off energies, around a few tens of GeV, but observations in this range have not been possible to date. The detection (or lack of) of a signal from known pulsars with MAGIC will shed some light on this problem. The suggestion that many of the EGRET unidentified sources are radio quiet pulsars can also be tested.

### 2.4. The Origin of Cosmic Rays

More than 90 years after their discovery, the origin of cosmic rays is still unclear. Despite shell-type supernova remnants have long been considered (mainly on grounds of energy budget arguments) the best candidates for the acceleration of the galactic cosmic ray nuclei, the direct proof of this hypothesis is still missing. For long, a signature of gamma rays from $\pi 0$ decay has been searched unsuccessfully in observations of several SNRs: the few positive detections of gamma signals can be well explained as the result of the inverse Compton interactions of high energy electrons with ambient photons. Deep field observations of promising candidates will be a part of the MAGIC observation campaigns.

### 2.5. Gamma Ray Bursts

Among new generation IACTs, MAGIC is the only one suited for the search for GRBs, due to its low threshold (30 GeV in the first phase), and to its capability for fast slewing. EGRET has detected 4 photons above 1 GeV from GRBs, and even a 18 GeV photon from GRB940217 (Hurley et al. 1994). Also, the MILAGRITO experiment may have detected TeV photons from another burst, GRB970417a. Extrapolations at MAGIC energies predict that one or two GRBs per year may be detectable by our instrument. A fast alert from satellite instruments (like the SWIFT mission to be launched end of this year) will be anyhow essential to allow MAGIC to catch any GRB before the end of the event. Apart from constraining GRB models, the determination of the light curve at GeV energies may be used, for instance, for tests of quantum gravity models, like the predicted Lorentz invariance deformation which would imply a small delay in the arrival of high energy photons.

## 3. The IACT Technique

The thickness of the atmosphere of the Earth (about 28 radiation lengths at sea level) prevents direct detection of high energy photons by ground-based instruments, even from the top of the highest mountains. However, the effects of the absorption of a gamma ray of a few giga electron Volt (GeV) or more in the atmosphere can be observed with a variety of detectors on a relatively large area on the ground around the hypothetical impact point of the photon. Gamma rays initiate electromagnetic showers of particles, a multiplicative process started by a $e\pm$ pair production in the electric field of an atmospheric nucleus.

The electrons and positrons give rise to secondary gammas via bremsstrahlung, which in turn produce more $e\pm$ pairs. The process goes on until the average energy of the shower particles drops below a critical value ($\simeq$ 100 MeV) beyond which the ionization and Compton scattering processes dominate. At this point the shower reaches its maximum development, and then

gradually extinguishes. Throughout the shower development, the electrons and positrons which travel faster than the speed of light in the air emit Cherenkov radiation. Cherenkov photons from a single shower arrive on Earth as a short pulse of a few nano seconds duration, and spread over a large area, roughly a disk of several hundred-meter diameter. This constitutes the basis of the most successful technique of ground-based gamma ray astronomy, that of Imaging Atmospheric Cherenkov Telescopes (IACTs).

After accounting for atmospheric extinction (due mainly to Rayleigh and Mie scattering), the spectrum of Cherenkov light at ground peaks in the near ultraviolet, around 330 nm. A telescope consisting of a large mirror and a camera of fast photodetectors can obtain a Cherenkov image of the shower above the fluctuations of the light of the night sky. Note that such device will be sensitive to any shower whose Cherenkov light pool contains the telescope, and hence the effective gamma ray collection area of an IACT is of the order of 10^5 m^2.

### 3.1. Background Rejection

One of the drawbacks of the IACT technique is the presence of a heavy background: air showers initiated by cosmic ray nuclei are detected by IACTs at a much higher rate than gammas, even during the observation of the strongest gamma sources. These hadronic showers are wider and more irregular than their electromagnetic counterparts, a fact that can be used in the off-line image analysis to reject most of them, following the methods first proposed by Hillas (1985). The shapes of shower images are parametrized by the momenta, up to second order, of the light distribution on the camera. An example of how this analysis is performed for the HEGRA CT1 telescope is, taken from Kranich (2002).

Besides, because of the isotropy of the cosmic radiation, the images of hadronic showers have random orientations, whereas the cigar-shaped images of gamma rays from a point-like source are oriented towards the source location on the camera, much like the tracks of shooting stars do. Consequently, when pointing the telescope towards a gamma ray source, the signal will appear as an access of shower images aligned with the other camera centre. This orientation parameter (ALPHA) is the only available tool to discriminate the otherwise irreducible background of cosmic ray electrons, which produce electromagnetic showers identical to those started by Gamma rays.

From the analysis of the images of showers surviving the background discrimination procedure, the energy spectrum of the observed source can be determined. The energy resolution of an IACT varies between 15 and 30% (statistical error), depending on energy, and is ultimately limited by the systematic error due to the uncertainty in the atmospheric absorption of Cherenkov light.

## 4. The MAGIC Gamma Telescope Data Set

**Abstract**:

Data are MC generated to simulate registration of high energy gamma particles in an atmospheric Cherenkov telescope.

## 4.1. Data Set Information

The data are MC generated (see below) to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. Cherenkov gamma telescope observes high energy gamma rays, taking advantage of the radiation emitted by charged particles produced inside the electromagnetic showers initiated by the gammas, and developing in the atmosphere. This Cherenkov radiation (of visible to UV wavelengths) leaks through the atmosphere and gets recorded in the detector, allowing reconstruction of the shower parameters.

The available information consists of pulses left by the incoming Cherenkov photons on the photomultiplier tubes, arranged in a plane, the camera. Depending on the energy of the primary gamma, a total of few hundreds to some 10000 Cherenkov photons get collected, in patterns (called the shower image), allowing to discriminate statistically those caused by primary gammas (signal) from the images of hadronic showers initiated by cosmic rays in the upper atmosphere (background).

Typically, the image of a shower after some pre-processing is an elongated cluster. Its long axis is oriented towards the camera centre if the shower axis is parallel to the telescope's optical axis, i.e., if the telescope axis is directed towards a point source. A principal component analysis is performed in the camera plane, which results in a correlation axis and defines an ellipse. If the depositions were distributed as a bivariate Gaussian, this would be an equi-density ellipse. The characteristic parameters of this ellipse (often called Hillas parameters) are among the image parameters that can be used for discrimination.

The energy depositions are typically asymmetric along the major axis, and this asymmetry can also be used in discrimination. There are, in addition, further discriminating characteristics, like the extent of the cluster in the image plane, or the total sum of depositions.

The data set was generated by a Monte Carlo program, Corsika, described in:
D. Heck et al., CORSIKA, A Monte Carlo code to simulate extensive air showers,
For schungszentrum Karlsruhe FZKA 6019 (1998).

The program was run with parameters allowing to observe events with energies down to below 50 GeV.

## 4.2. Attribute Information

1. fLength: major axis of ellipse [mm]
2. fWidth: minor axis of ellipse [mm]
3. fSize: 10-log of sum of content of all pixels [in #phot]
4. fConc: ratio of sum of two highest pixels over fSize [ratio]
5. fConc1: ratio of highest pixel over fSize [ratio]
6. fAsym: distance from highest pixel to centre, projected onto major axis [mm]
7. fM3Long: 3rd root of third moment along major axis [mm]
8. fM3Trans: 3rd root of third moment along minor axis [mm]
9. fAlpha: angle of major axis with vector to origin [deg]
10. fDist: distance from origin to center of ellipse [mm]

11. class: g, h # gamma (signal), hadron (background)

    g = gamma (signal): 12332
    h = hadron (background): 6688

For technical reasons, the number of h events is underestimated. In the real data, the h class represents most of the events.

The simple classification accuracy is not meaningful for this data, since classifying a background event as signal is worse than classifying a signal event as background. For comparison of different classifiers an ROC curve must be used. The relevant points on this curve are those, where the probability of accepting a background event as signal is below one of the following thresholds: 0.01, 0.02, 0.05, 0.1, 0.2 depending on the required quality of the sample of the accepted events for different experiments.

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 19020 | **Area:** | Physical |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 11 | **Date Donated** | 2007-05-01 |
| **Associated Tasks:** | Classification | **Missing Values?** | No | **Number of Web Hits:** | 138776 |

## 5. Classification Algorithms

### 5.1. Introduction to Classification

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

For example, spam detection in email service providers can be identified as a classification problem. This is s binary classification since there are only 2 classes as spam and not spam. A classifier utilizes some training data to understand how given input variables relate to the class. In this case, known spam and non-spam emails must be used as the training data. When the classifier is trained accurately, it can be used to detect an unknown email.

Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing etc.

There are two types of learners in classification as lazy learners and eager learners.

#### 5.1.1. **Lazy learners**

Lazy learners simply store the training data and wait until a testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting.

*Ex. k-nearest neighbour, Case-based reasoning*

#### 5.1.2. **Eager learners**

Eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Due to the model construction, eager learners take a long time for train and less time to predict.

*Ex. Decision Tree, Naive Bayes, Artificial Neural Networks*

There is a lot of classification algorithms available now, but it is not possible to conclude which one is superior to other. It depends on the application and nature of available data set. For example, if the classes are linearly separable, the linear classifiers like Logistic regression, Fisher's linear discriminant can outperform sophisticated models and vice versa.

## 6. Classification Algorithms Applied

### 6.1. XGBoost (Gradient Boost Classifier)

The beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

It's no wonder then that CERN recognized it as the best approach to classify signals from the Large Hadron Collider. This challenge posed by CERN required a solution that would be scalable to process data being generated at the rate of 3 petabytes per year and effectively distinguish an extremely rare signal from background noises in a complex physical process. XGBoost emerged as the most useful, straight forward, and robust solution.

XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models.

The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. **Bagging and boosting are two widely used ensemble learners**. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.

#### 6.1.1. Bagging

While decision trees are one of the most easily interpretable models, they exhibit highly variable behaviour. Consider a single training dataset that we randomly split into two parts. Now, let's use each part to train a decision tree to obtain two models.

When we fit both these models, they would yield different results. Decision trees are said to be associated with high variance due to this behaviour. Bagging or boosting aggregation helps to reduce the variance in any learner. Several decision trees which are generated in parallel, form the base learners of bagging technique. Data sampled with replacement is fed to these learners for training. The final prediction is the averaged output from all the learners.

### 6.1.2. Boosting

In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.

The base learners in boosting are weak learners in which the bias is high, and the predictive power is just a tad better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance.

## 6.2. Random Forest Classifier

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is that it can be used for both classification and regression problems, which form most current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning.

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn provides a great tool for this that measures a feature's importance by looking at how much the tree nodes that use that feature reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so the sum of all importance is equal to one.

If you don’t know how a decision tree works or what a leaf or node is, here is a good description from Wikipedia: '"In a decision tree each internal node represents a 'test' on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). A node that has no children is a leaf.'"

By looking at the feature importance you can decide which features to possibly drop because they don’t contribute enough (or sometimes nothing at all) to the prediction process.

This is important because a general rule in machine learning is that the more features you have the more likely your model will suffer from overfitting and vice versa.

### 6.3. Stochastic Gradient Descent Classifier

Stochastic Gradient Descent (SGD) is a simple, yet efficient optimization algorithm used to find the values of parameters/coefficients of functions that minimize a cost function. In other words, it is used for discriminative learning of linear classifiers under convex loss functions such as SVM and Logistic regression. It has been successfully applied to large-scale datasets because the update to the coefficients is performed for each training instance, rather than at the end of instances.

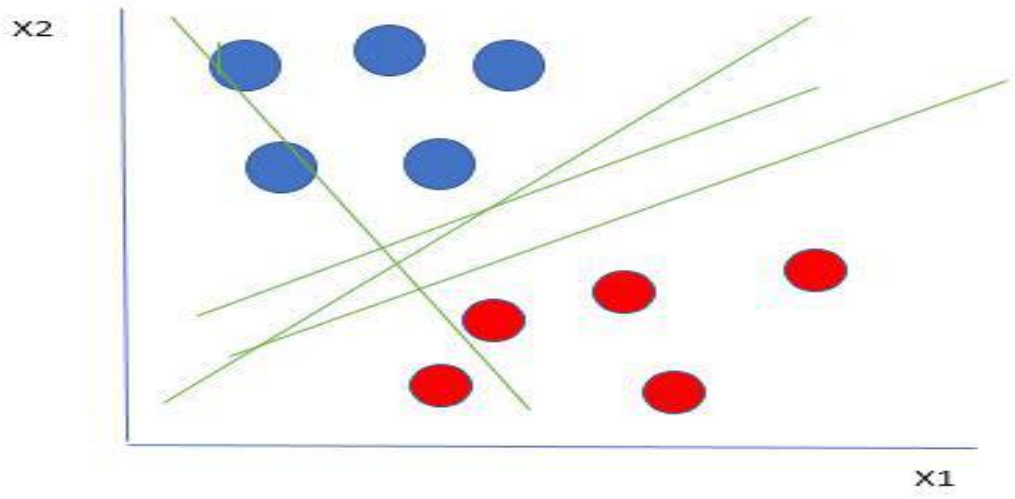
Stochastic Gradient Descent (SGD) classifier basically implements a plain SGD learning routine supporting various loss functions and penalties for classification. Scikit-learn provides **SGD Classifier** module to implement SGD classification.

### 6.4. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification.

The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Let's consider two independent variables x1, x2 and one dependent variable which is either a blue circle or a red circle.

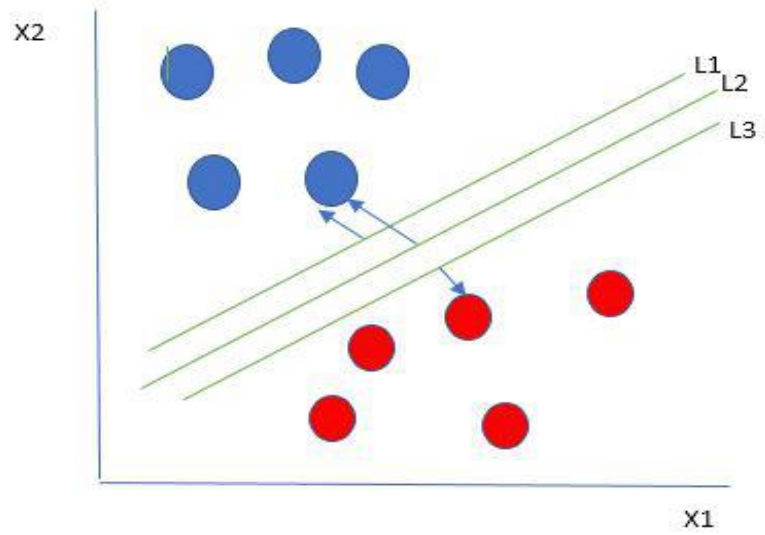



*Linearly Separable Data points*

From the figure above its very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregates our data points or does a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points.

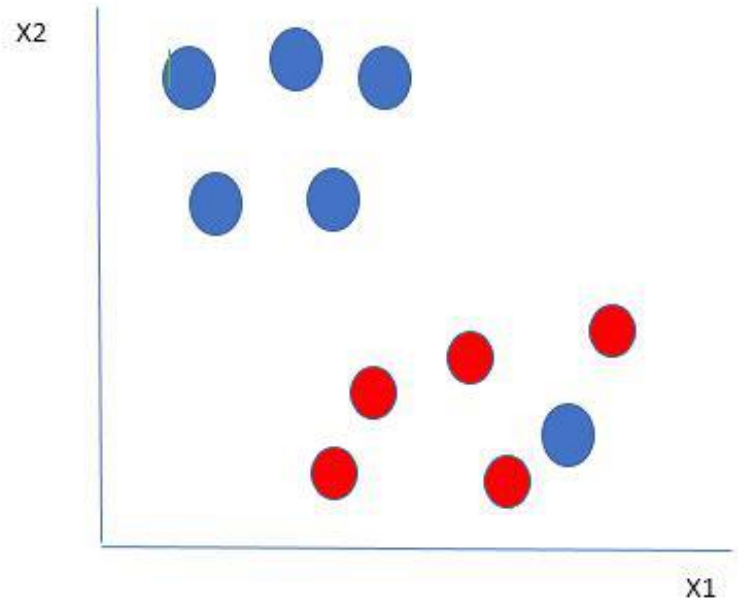
### 6.4.1. **Selecting the best hyper-plane:**

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

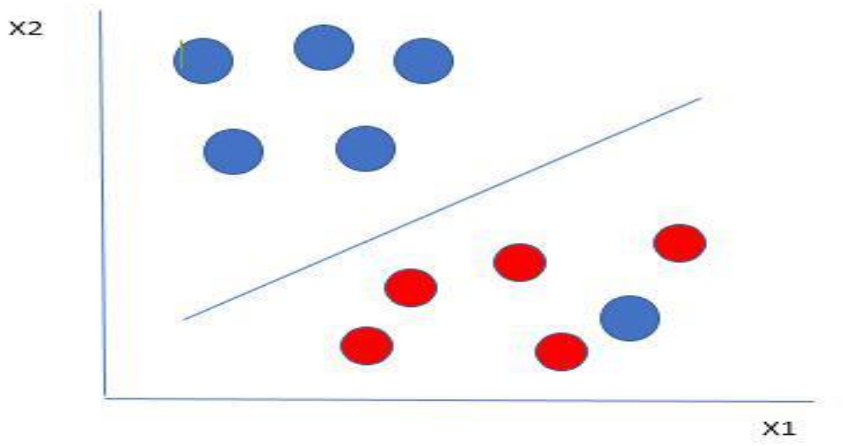



So, we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane/hard margin. So, from the above figure, we choose L2.

Let's consider a scenario like shown below

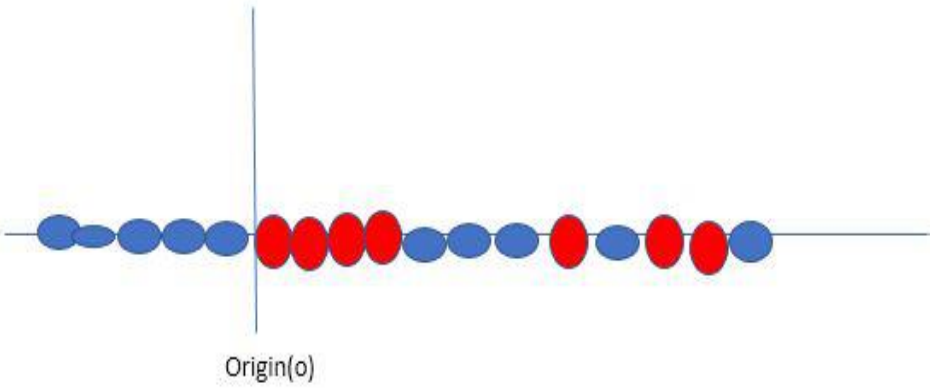



Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It's simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

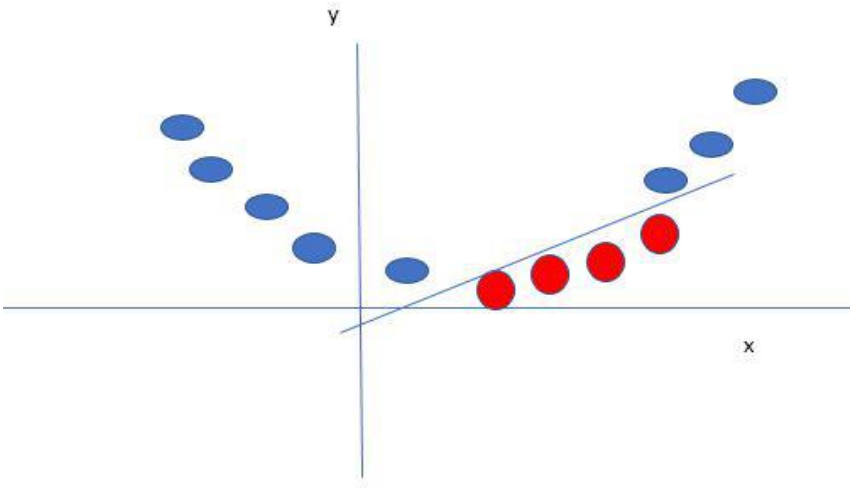



So, in this type of data points what SVM does is, it finds maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these type of cases are called soft margin. When there is a soft margin to the data set, the SVM tries to minimize *(1/margin+∧(∑penalty))*. Hinge loss is a commonly used penalty. If no violations no hinge loss. If violations hinge loss proportional to the distance of violation.

Till now, we were talking about linearly separable data (the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?




Say, our data is like shown in the figure above. SVM solves this by creating a new variable using a kernel. We call a point $x_i$ on the line, and we create a new variable $y_i$ as a function of distance from origin o.so if we plot this, we get something like as shown below




In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as kernel.

**SVM Kernel:**

The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, i.e., it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel, it does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

### 6.5. Decision Tree Classification

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome**.**

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed based on features of the given dataset.

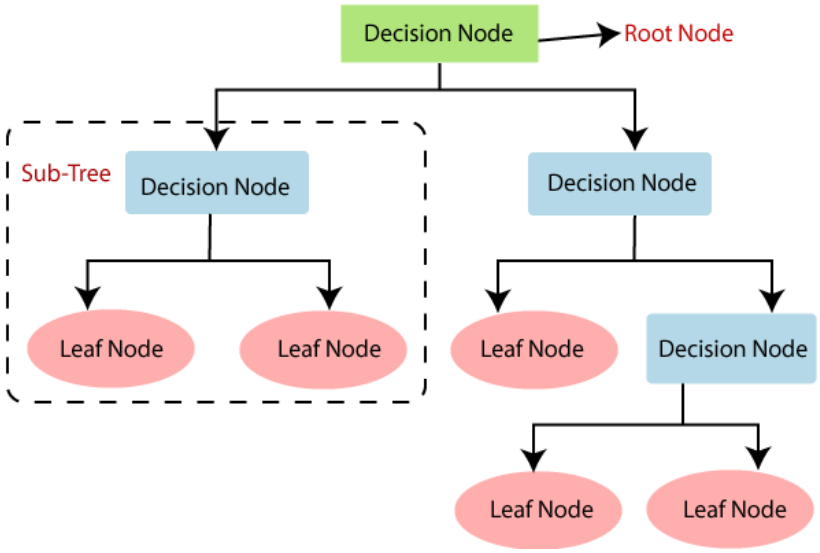
*It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.*

It is called a decision tree because, like a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

To build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

Below diagram explains the general structure of a decision tree:




### 6.5.1. Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

### 6.5.2. Working:

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and based on the comparison, follows the branch, and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.

- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

### 6.5.3. Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

#### 6.5.3.1 Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy (each feature)

**Entropy:**

Entropy is a metric to measure the impurity in each attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

#### 6.5.3.2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.

- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula: Gini Index= $1- \sum_j P_j^2$

### 6.5.4. Pruning: Getting an Optimal Decision tree

*Pruning is a process of deleting the unnecessary nodes from a tree to get the optimal decision tree.*

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of trees pruning technology used:

a. Cost Complexity Pruning
b. Reduced Error Pruning.

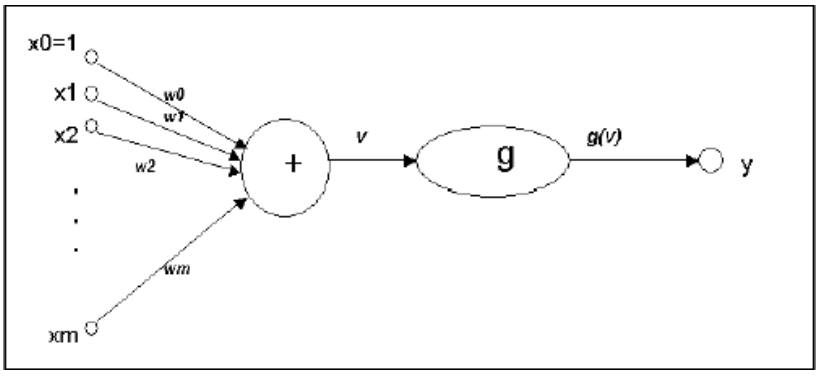
## 6.6. Neural Network

Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. They process records one at a time and learn by comparing their classification of the record (i.e., largely arbitrary) with the known actual classification of the record. The errors from the initial classification of the first record are fed back into the network and used to modify the networks algorithm for further iterations.

A neuron in an artificial neural network is

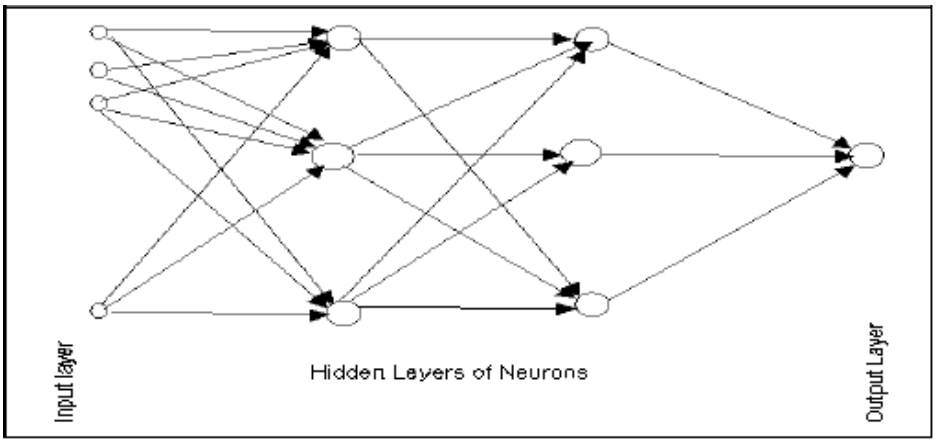
1. A set of input values (xi) and associated weights (wi).

2. A function (g) that sums the weights and maps the results to an output (y).




Neurons are organized into layers: input, hidden and output. The input layer is composed not of full neurons, but rather consists simply of the record's values that are inputs to the next layer of neurons. The next layer is the hidden layer. Several hidden layers can exist in one neural network. The final layer is the output layer, where there is one node for each class. A single sweep forward through the network

results in the assignment of a value to each output node, and the record is assigned to the class node with the highest value.




### 6.6.1. Training an Artificial Neural Network

In the training phase, the correct class for each record is known (termed supervised training), and the output nodes can be assigned correct values -- 1 for the node corresponding to the correct class, and 0 for the others. (In practice, better results have been found using values of 0.9 and 0.1, respectively.) It is thus possible to compare the network's calculated values for the output nodes to these correct values and calculate an error term for each node (the Delta rule). These error terms are then used to adjust the weights in the hidden layers so that, hopefully, during the next iteration the output values will be closer to the correct values.

### 6.6.2. The Iterative Learning Process

A key feature of neural networks is an iterative learning process in which records (rows) are presented to the network one at a time, and the weights associated with the input values are adjusted each time. After all cases are presented, the process is often repeated. During this learning phase, the network trains by adjusting the weights to predict the correct class label of input samples. Advantages of neural networks include their high tolerance to noisy data, as well as their ability to classify patterns on which they have not been trained. The most popular neural network algorithm is the back-propagation algorithm proposed in the 1980s.

Once a network has been structured for a particular application, that network is ready to be trained. To start this process, the initial weights (described in the next section) are chosen randomly. Then the training (learning) begins.

The network processes the records in the Training Set one at a time, using the weights and functions in the hidden layers, then compares the resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights for application to the next record. This process occurs repeatedly as the weights are tweaked. During the training of a network, the same set of data is processed many times as the connection weights are continually refined.

**Note** that some networks never learn. This could be because the input data does not contain the specific information from which the desired output is derived. Networks also will not converge if there is not enough data to enable complete learning. Ideally, there should be enough data available to create a Validation Set.

### 6.6.3. Feedforward, Back-Propagation

The feedforward, back-propagation architecture was developed in the early 1970s by several independent sources (Werbor; Parker; Rumelhart, Hinton, and Williams). This independent co-development was the result of a proliferation of articles and talks at various conferences that stimulated the entire industry. Currently, this synergistically developed back-propagation architecture is the most popular model for complex, multi-layered networks. Its greatest strength is in non-linear solutions to ill-defined problems.

The typical back-propagation network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there are just one or two. Some studies have shown that the total number of layers needed to solve problems of any complexity is five (one input layer, three hidden layers and an output layer). Each layer is fully connected to the succeeding layer.

The training process normally uses some variant of the Delta Rule, which starts with the calculated difference between the actual outputs and the desired outputs. Using this error, connection weights are increased in proportion to the error times, which are a scaling factor for global accuracy. This means that the inputs, the output, and the desired output all must be present at the same processing element. The most complex part of this algorithm is determining which input contributed the most to an incorrect output and how must the input be modified to correct the error. (An inactive node would not contribute to the error and would have no need to change its weights.)

To solve this problem, training inputs are applied to the input layer of the network, and desired outputs are compared at the output layer. During the learning process, a forward sweep is made through the network, and the output of each element is computed by layer. The difference between the output of the final layer and the desired output is backpropagated to the previous layer(s), usually modified by the derivative of the transfer function. The connection weights are normally adjusted using the Delta Rule. This process proceeds for the previous layer(s) until the input layer is reached.

### 6.6.4. Structuring the Network

The number of layers and the number of processing elements per layer are important decisions. To a feedforward, back-propagation topology, these parameters are also the most ethereal -- they are the art of the network designer. There is no quantifiable answer to the layout of the network for any application. There are only general rules picked up over time and followed by most researchers and engineers applying while this architecture to their problems.

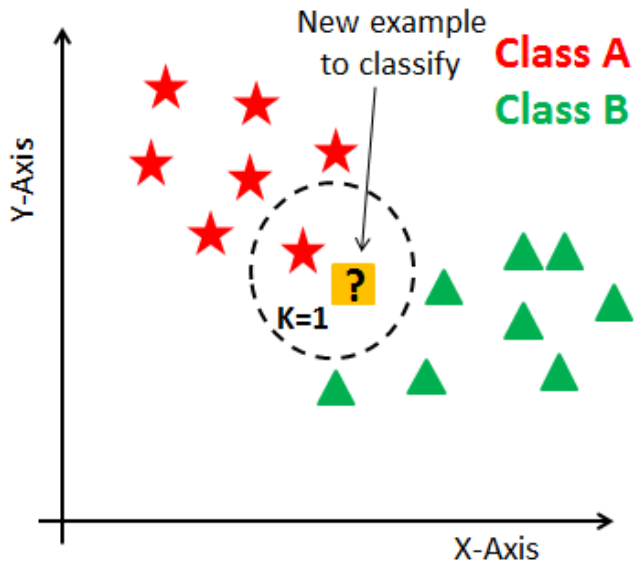
Rule One: As the complexity in the relationship between the input data and the desired output increases, the number of the processing elements in the hidden layer should also increase.

Rule Two: If the process being modelled is separable into multiple stages, then additional hidden layer(s) may be required. If the process is not separable into stages, then additional layers may simply enable memorization of the training set, and not a true general solution.

Rule Three: The amount of Training Set available sets an upper bound for the number of processing elements in the hidden layer(s). To calculate this upper bound, use the number of cases in the Training Set and divide that number by the sum of the number of nodes in the input and output layers in the network. Then divide that result again by a scaling factor between five and ten. Larger scaling factors are used for relatively less noisy data. If too many artificial neurons are used the Training Set will be memorized, not generalized, and the network will be useless on new data sets.

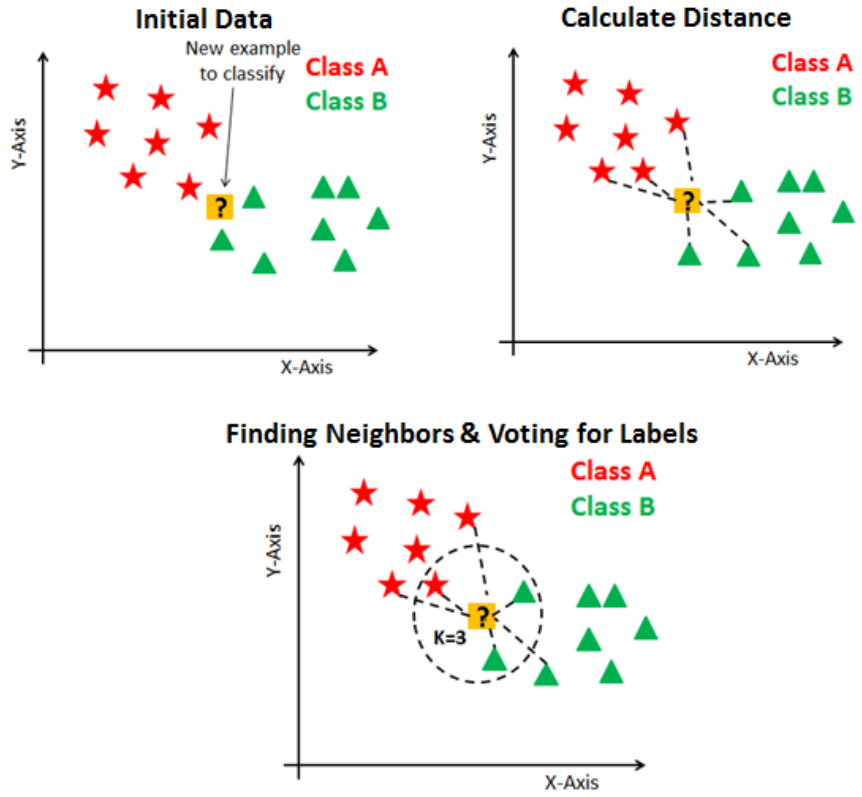
### 6.7. K Neighbors Classifier

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.




Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels




### 6.7.1. Eager Vs. Lazy Learners

**Eager learners** mean when given training points will construct a generalized model before performing prediction on given new points to classify. You can think of such learners as being ready, active, and eager to classify unobserved data points.

**Lazy Learning** means there is no need for learning or training of the model and all the data points used at the time of prediction. Lazy learners wait until the last minute before classifying any data point. Lazy learner stores merely the training dataset and waits until classification needs to perform. Only when it sees the test tuple does it perform generalization to classify the tuple based on its similarity to the stored training tuples. Unlike eager learning methods, lazy learners do less work in the training phase and more work in the testing phase to make a classification. Lazy learners are also known as instance-based learners because lazy learners store the training points or instances, and all learning is based on instances.

### 6.7.2. Curse of Dimensionality

KNN performs better with a lower number of features than many features. You can say that when the number of features increases than it requires more data. Increase in dimension also leads to the problem of overfitting. To avoid overfitting, the needed data will need to grow exponentially as you increase the number of dimensions. This problem of higher dimension is known as the Curse of Dimensionality.

To deal with the problem of the curse of dimensionality, you need to perform principal component analysis before applying any machine learning algorithm, or you can also use feature selection approach. Research has shown that in large dimension Euclidean distance is not useful anymore. Therefore, you can prefer other measures such as cosine similarity, which get decidedly less affected by high dimension.

### 6.7.3. **Algorithm**

Let m be the number of training data samples. Let p be an unknown point.

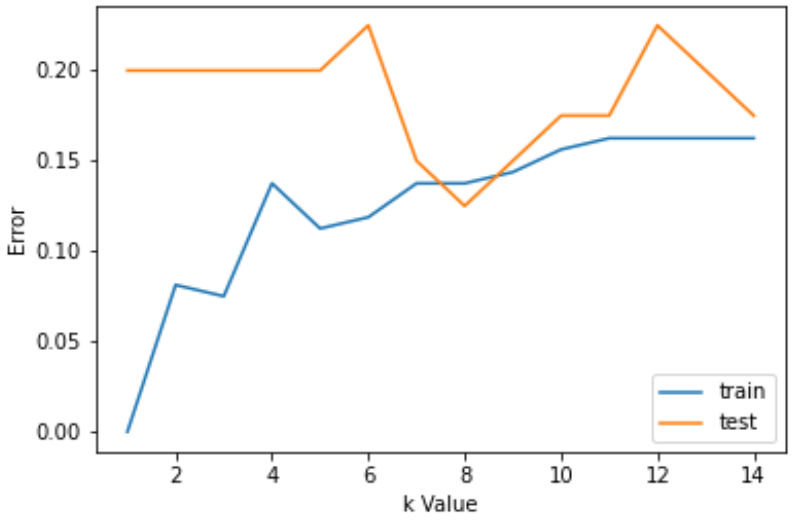
1. Store the training samples in an array of data points arr []. This means each element of this array represents a tuple (x, y).
2. for i=0 to m:
3. Calculate Euclidean distance d(arr[i], p).
4. Make set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point.
5. Return the majority label among S.

K can be kept as an odd number so that we can calculate a clear majority in the case where only two groups are possible (e.g., Red/Blue). With increasing K, we get smoother, more defined boundaries across different classifications. Also, the accuracy of the above classifier increases as we increase the number of data points in the training set.

### 6.7.4. How to choose the value for K?

K is a crucial parameter in the KNN algorithm. Some suggestions for choosing K Value are:

1. Using error curves: The figure below shows error curves for different values of K for training and test data.
2. At low K values, there is overfitting of data/high variance. Therefore, test error is high and train error is low. At K=1 in train data, the error is always zero, because the nearest neighbor to that point is that point itself. Therefore, though training error is low test error is high at lower K values. This is called overfitting. As we increase the value for K, the test error is reduced.
3. But after a certain K value, bias/ underfitting is introduced and test error goes high. So, we can say initially test data error is high (due to variance) then it goes low and stabilizes and with further increase in K value, it again increases (due to bias). The K value when test error stabilizes and is low is considered as optimal value for K. From the above error curve, we can choose K=8 for our KNN algorithm implementation.
4. Also, domain knowledge is very useful in choosing the K value.
5. K value should be odd while considering binary(two-class) classification.




### 6.7.5. Required Data Preparation:

1. Data Scaling: To locate the data point in multidimensional feature space, it would be helpful if all features are on the same scale. Hence normalization or standardization of data will help.
2. Dimensionality Reduction: KNN may not work well if there are too many features. Hence dimensionality reduction techniques like feature selection, principal component analysis can be implemented.
3. Missing value treatment: If out of M features one feature data is missing for a particular example in the training set, then we cannot locate or calculate distance from that point. Therefore, deleting that row or imputation is required.

## 6.8. Naïve Bayes Classifier

### 6.8.1. Bayes Theorem

It is a theorem that works on conditional probability. Conditional probability is the probability that something will happen, given that something else has already occurred. The conditional probability can give us the probability of an event using its prior knowledge.

### 6.8.2. Conditional probability:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

*Conditional Probability*

Where,
P(A): The probability of hypothesis H being true. This is known as the prior probability.
P(B): The probability of the evidence.
P(A|B): The probability of the evidence given that hypothesis is true.
P(B|A): The probability of the hypothesis given that the evidence is true.

### 6.8.3. Naive Bayes Classifier

A classifier is a machine learning model segregating different objects based on certain features of variables.

It is a kind of classifier that works on the Bayes theorem. Prediction of membership probabilities is made for every class such as the probability of data points associated with a particular class.

The class having maximum probability is appraised as the most suitable class. This is also referred to as Maximum A Posteriori (MAP).

- The MAP for a hypothesis is:
  - $MAP\ (H)$ = max $P((H|E))$
  - $MAP\ (H)$ = max $P((H|E) * (P(H))\ /P(E))$
  - $MAP\ (H)$ = max$(P(E|H) * P(H))$
  - $P\ (E)$ is evidence probability, and it is used to normalize the result. The result will not be affected by removing $P(E)$.

Naive Bayes classifiers conclude that all the variables or features are not related to each other. The Existence or absence of a variable does not impact the existence or absence of any other variable. For example,

- Fruit may be observed to be an apple if it is red, round, and about 4″ in diameter.
- In this case also even if all the features are interrelated to each other, an naive bayes classifier will observe all of these independently contributing to the probability that the fruit is an apple.

We experiment with the hypothesis in real datasets, given multiple features. So, computation becomes complex.

### 6.8.4. Types of Naive Bayes Algorithms

1. **Gaussian Naïve Bayes:**

   When characteristic values are continuous in nature then an assumption is made that the values linked with each class are dispersed according to Gaussian that is Normal Distribution.

2. **Multinomial Naïve Bayes:**

   Multinomial Naive Bayes is favoured to use on data that is multinomial distributed. It is widely used in text classification in NLP. Each event in text classification constitutes the presence of a word in a document.

3. **Bernoulli Naïve Bayes:**
   When data is dispensed according to the multivariate Bernoulli distributions then Bernoulli Naive Bayes is used. That means there exist multiple features but each one is assumed to contain a binary value. So, it requires features to be binary-valued.

## 6.9. Perceptron

A perceptron model, in Machine Learning, is a supervised learning algorithm of binary classifiers. A single neuron, the perceptron model detects whether any function is an input or not and classifies them in either of the classes.

Representing a biological neuron in the human brain, the perceptron model or simply a perceptron acts as an artificial neuron that performs human-like brain functions. A linear ML algorithm, the perceptron conducts binary classification or two-class categorization and enables neurons to learn and register information procured from the inputs.

This model uses a hyperplane line that classifies two inputs and classifies them based on the 2 classes that a machine learns, thus implying that the perceptron model is a linear classification model. Invented by Frank Rosenblatt in 1957, the perceptron model is a vital element of Machine Learning as ML is recognized for its classification purposes and mechanism.

There are 4 constituents of a perceptron model. They are as follows-

1. Input values
2. Weights and bias
3. Net sum
4. Activation function

The perceptron model enables machines to automatically learn coefficients of weight which helps in classifying the inputs. Also recognized as the Linear Binary Classifier, the perceptron model is extremely efficient and helpful in arranging the input data and classifying the same in different classes.

### 6.9.1. Understanding the Perceptron

The simplest variant of artificial neuron networks, the perceptron model resembles a biological neuron that simply helps in the linear binary classification with the help of a hyperplane line.

There are 2 types of perceptron models-

1. **Single Layer Perceptron-** The Single Layer perceptron is defined by its ability to linearly classify inputs. This means that this kind of model only utilizes a single hyperplane line and classifies the inputs as per the learned weights beforehand.

2. **Multi-Layer Perceptron-** The Multi-Layer Perceptron is defined by its ability to use layers while classifying inputs. This type is a high processing algorithm that allows machines to classify inputs using various more than one layer at the same time.

The working of the model is based on the Perceptron Learning Rule that implies that the algorithm is enabled to automatically learn respective coefficients of weights that designate several inputs.

The perceptron model registers inputs with the machine and allots them with certain weights as per the coefficients that lead a particular input into a specific class. This is decided based on the final value derived by calculating the net sum and activation function at the end stages.

Let us now go through a step-by-step procedure to understand the way the perceptron model operates.

0. Enter bits of information that are supposed to serve as inputs in the first layer (Input Value).
1. All weights (pre-learned coefficients) and input values will be multiplied. The multiplied values of all input values will be added.
2. The bias value will shift to the final stage (activation function/output result).
3. The weighted input will proceed to the stage of the activation function. The bias value will be now added.
4. The value procured will be the output value that will determine if the output will be released or not.

The perceptron algorithm, using the Heaviside activation function is summarised as follows-

$f(z) = \{1 \text{ if } xTw+b > 0$
$= \{0 \text{ otherwise}$

The Input value of the model consists of various artificial neurons in artificial intelligence that facilitate the entry of data into the system or machine.

When the inputs are registered in the machine, the perceptron algorithm primarily applies the already learned value of weight (dimension or strength of the connection between data units). These weights are then multiplied with the input values and headed to the net sum (total value).

Ultimately, the input value proceeds to the activation function where output is released or scrapped out. The activation function (weighted sum added with bias) in the final stage is important for determining if an input's value is greater than 0.

The process that enables the perceptron model to conduct mathematical operations for converting input into output is called training. As the process of training is implemented in the working of the perceptron model wherein machines are made fully capable of calculating output values even without being fed with input values.

The process of training involves feeding machines with historic data to prepare them for the future and instil predictive patterns. Based on artificial neural networks that tend to imitate the human brain, the perceptron model works along the lines of machine learning as it continuously interprets data and produces qualitative patterns.

**"It's essentially finding patterns in the training data and generating weights that will produce useful output by applying these patterns to new data."-Perceptron Model**

### 6.9.2. **Significance of the Perceptron model**

The Perceptron Model in Machine Learning is a supervised learning algorithm that focuses on the linear binary classification of inputs. This algorithm, when it was primarily being worked upon, was intended to facilitate image recognition in machines.

The model was considered a benchmark innovation for the development of Artificial Intelligence technology as it had the capability to strengthen the existing Machine Learning algorithms and even generate more advanced ones.

As it primarily worked and its popularity boosted up in the initial days, a lot of hope and positivity was attached to this innovation. However, it was soon that the infrastructural limitations were exposed, and it was thus perceived to be a long way before perceptron could be easily implemented.

A learning algorithm that self-arranges a network of artificial neurons to incorporate desired behaviours, the significance of the perceptron model enables machines to work efficiently in terms of binary classification.

While the single layer perceptrons can only classify inputs into classes with a single layer, the multi-layer perceptron enables the model to conduct a classification of inputs with the help of more than one layer, making it suitable for more advanced and complex inputs.

A supervised learning algorithm, like the perceptron model, is the most sought-after algorithm that prevails in the field of Machine Learning. Prevalent in the field of data analytics, the perceptron model initiates binary classification and leads to problem-solving when it comes to bits of data.

## 7. Accuracy Score

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy= Number of correct predictions / Total number of predictions

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

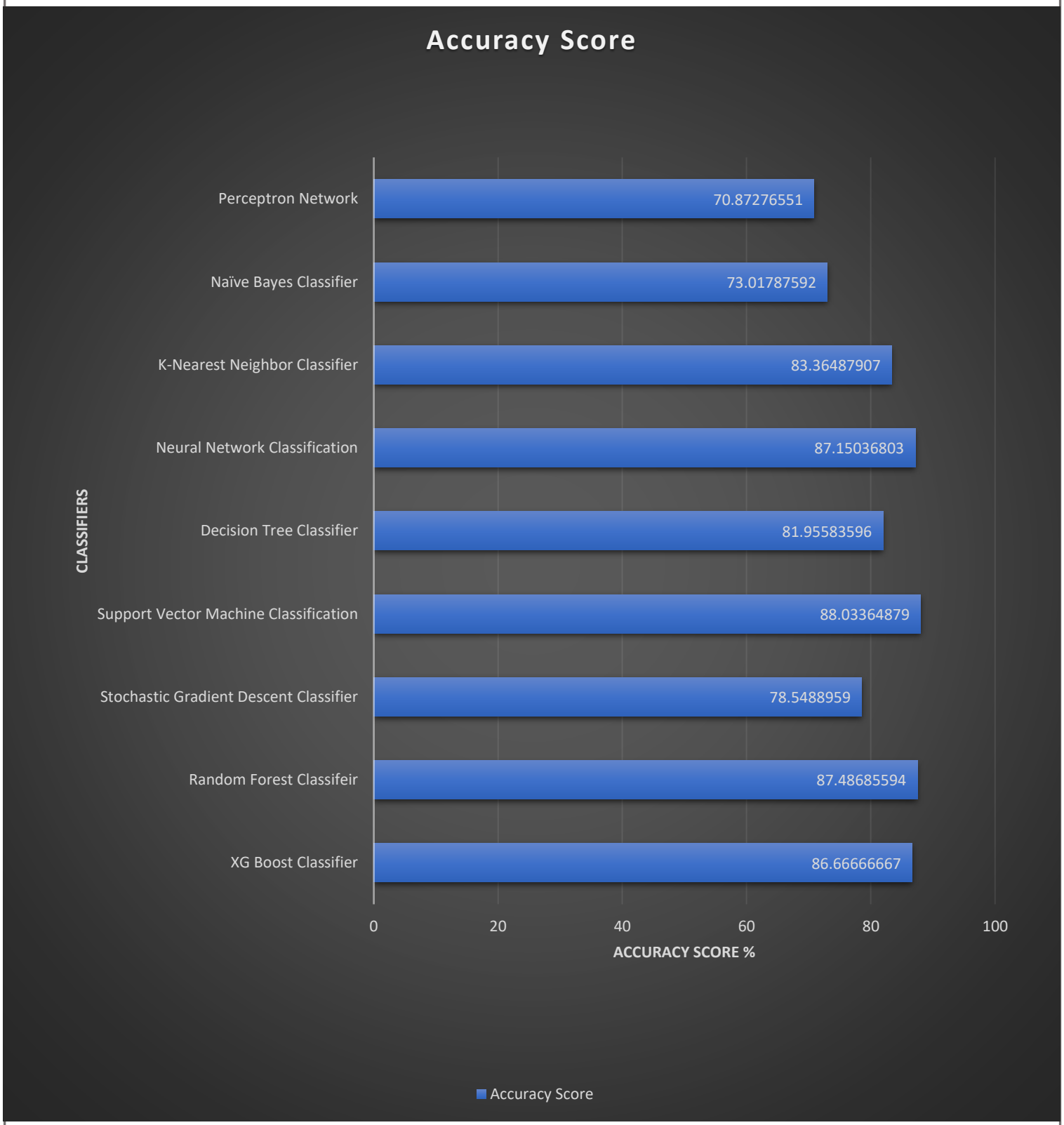
Accuracy=(TP+TN) / (TP+TN+FP+FN)

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

### 7.1. Accuracy Score of Implemented Models

a. XGB Classifier:
   Accuracy Score=> 86.66666666666667
b. Random Forest Classifier:
   Accuracy Score=> 87.48685594111461
c. Stochastic Gradient Descent Classifier:
   Accuracy Score=> 78.54889589905363
d. Support Vector Machine:
   Accuracy Score=> 88.03364879074658
e. Decision Tree Classifier:
   Accuracy Score=> 81.9558359621451
f. Neural Network Classifier:
   Accuracy Score=> 87.1503680336487,
g. K-Nearest Neighbor Classifier:
   Accuracy Score=> 83.36487907465825
h. Naive Bayes Classifier:
   Accuracy Score=> 73.01787592008412
i. Perceptron:
   Accuracy Score=> 70.87276550998949

## 7.2. Visualization:



## 8. Result

Support H=Vector Machine has the highest accuracy score of 88.03 % and Naïve Bayes Classifier has the lowest accuracy score of 73.02% over the Dataset.

## 9. Conclusion

The dataset was miraculously free of null values which made out classification hassle free. From the GRAPH represented above, I can confidently say that Support Vector Machine has proved to be superior classifier for this dataset.

## 10. References


10.1. Classification methods for Cherenkov telescopes images on a pixel-by-pixel base C. MALAGON 1, J.A. BARRIO2, D. NIETO 2, R. DE LOS REYES 2 1*Dpto. Ingenier´ıa Informatica, Univ. Antonio de Nebrija, 28040 Madrid, Spain.* 2*Dpto. F´ısica Atomica, Univ. Complutense, 28040 Madrid, Spain*

10.2. Gamma-Hadron Classification for the Ground Based Atmospheric Cherenkov Telescope MACE By MRADUL SHARMA, Bhabha Atomic Research Centre, INDIA

10.3. H.E.S.S. and MAGIC observations of a sudden cessation of a very-high-energy -ray flare in PKS1510-089

10.4. Observation of the Gamma-Ray Binary HESS J0632+057 with the H.E.S.S., MAGIC, and VERITAS Telescopes

10.5. Spatial likelihood analysis for MAGIC telescope data from instrument response modelling to spectral extraction Vovk1, M. Strzys1, and C. Fruck2,1Max-Planck-Institut für Physik, Föhringer Ring 6, 80805 Munich, Germany

10.6. Toward a Public MAGIC Gamma-Ray Telescope Legacy Data Portal | M. Doroa, C. Nigroe, E. Prandinia, A. Tramacereb, M. Delfinoc; d, J. Delgadoc; d, E. do Soutoc, L. Jouvinc, J. Ricoc for the MAGIC Collaboration

10.7. Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope | R.K. Bock a. A. Chilingarian | b. M. Gaug | c. F. Hakl | d. T. Hengstebeck

10.8. Softening Splits in Decision Trees Using Simulated Annealing | Jakub Dvorak and Petr Savick´y Institute of Computer Science, Academy of Sciences of the Czech Republic

10.9. Discernibility Concept in Classification Problems by *Zacharias N. Voulgaris*

10.10. EXPERIMENTAL STUDY OF LEAF CONFIDENCES FOR RANDOM FORESTS | Petr Savick´y and Emil Kotrˇc (*Key words*: Decision trees, random forests, weights, leaf confidences.) *COMPSTAT 2004 section*: Neural networks and machine learning.