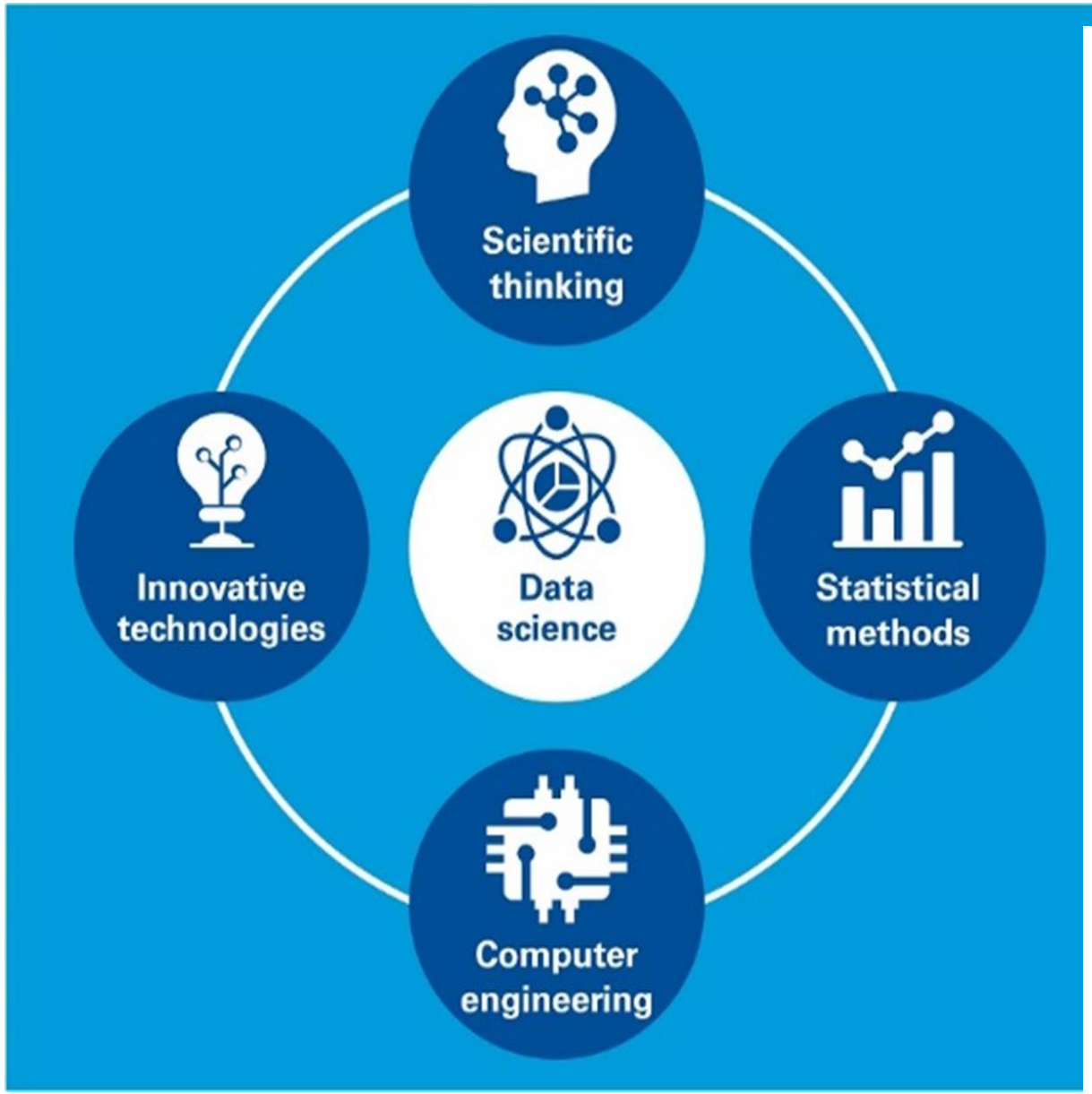
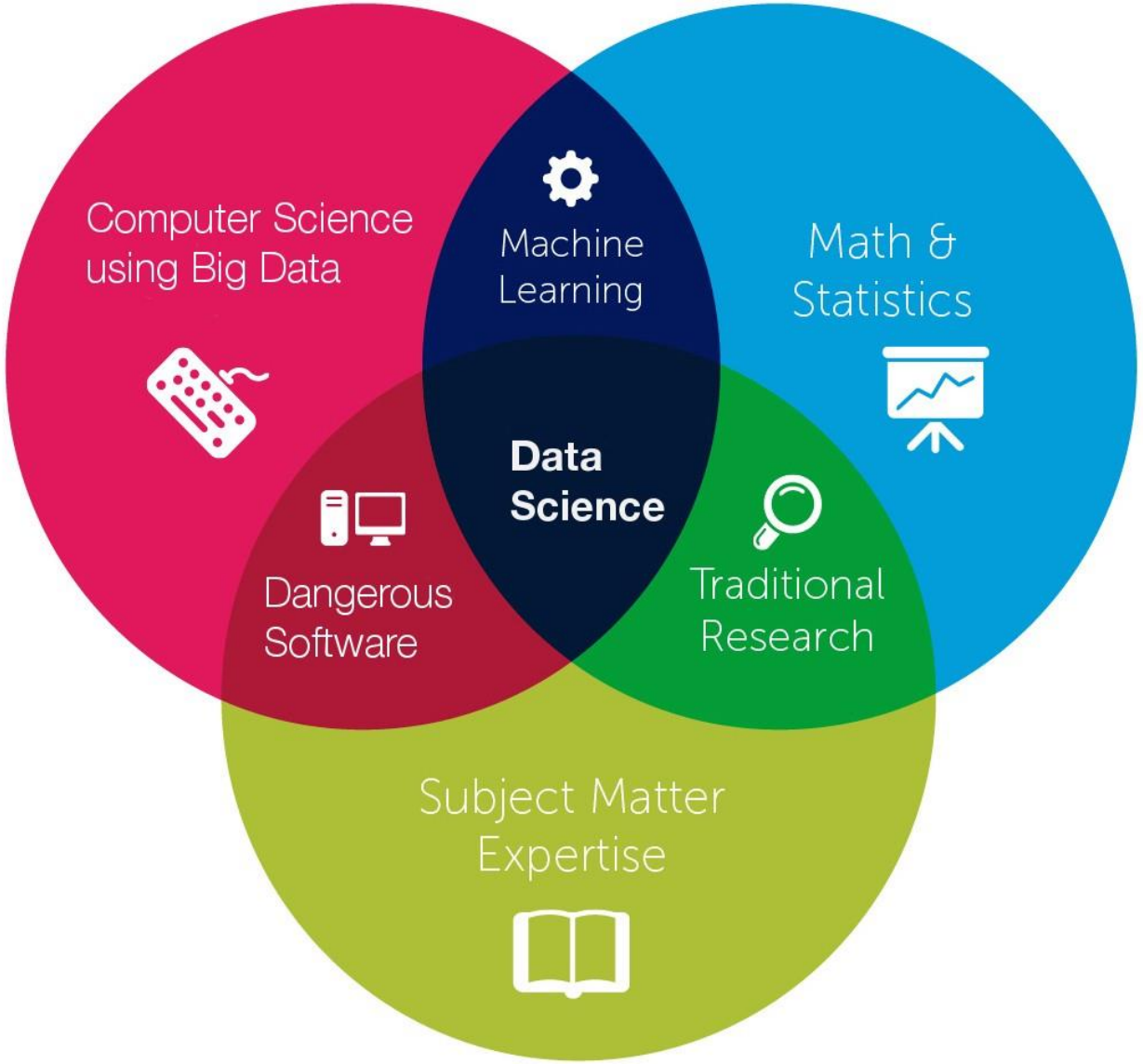
# Industry Orientation on Emerging Trends/ Technologies


Scientific thinking
Innovative technologies
Data science
Statistical methods
Computer engineering

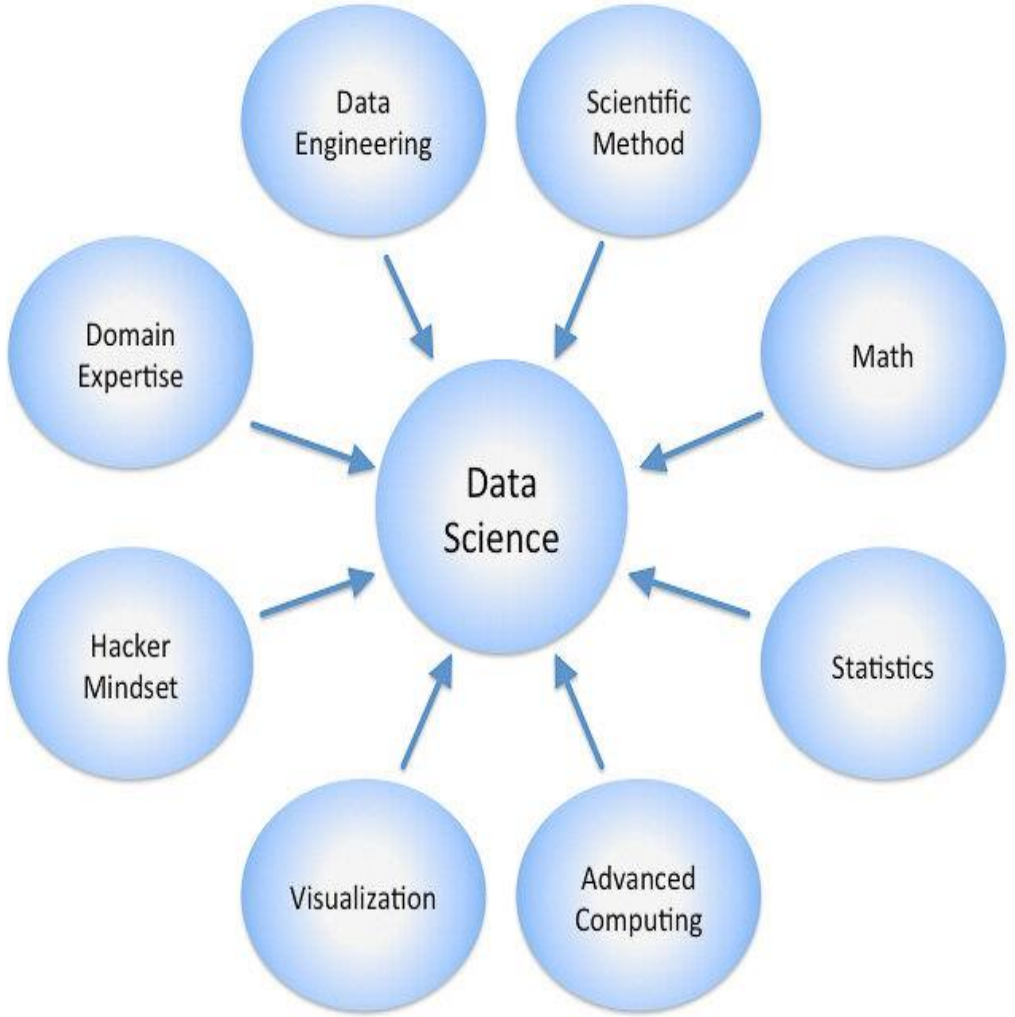

Computer Science using Big Data
Machine Learning
Math & Statistics
Data Science
Dangerous Software
Traditional Research
Subject Matter Expertise

# Data Science

- **Data science** is a combined practice of **programming skills, mathematics, statistics**, and **domain expertise** to retrieve meaningful insights from the given data.
- Data is one of the major features of every firm as it helps business leaders to make informed decisions based on numbers, trends, and facts.
- As per PayScale, on average, a data scientist can earn ₹698,400 per year in India and $123,000 per year in the US.
- Data Science is important because it helps businesses in many ways as follows:
- Enabling management and officers to make data-driven decisions
- It helps organizations to identify their client in a better way

# The Pillars of Data Science Expertise

- While data scientists often come from many different educational and work experience backgrounds, mos should be strong in, or in an ideal case be experts in fou fundamental areas. In no particular order of priority o importance, these are:
- Business/Domain
- Mathematics (includes statistics and probability)
- Computer science (e.g., software/data architecture an engineering)
- Communication (both written and verbal)

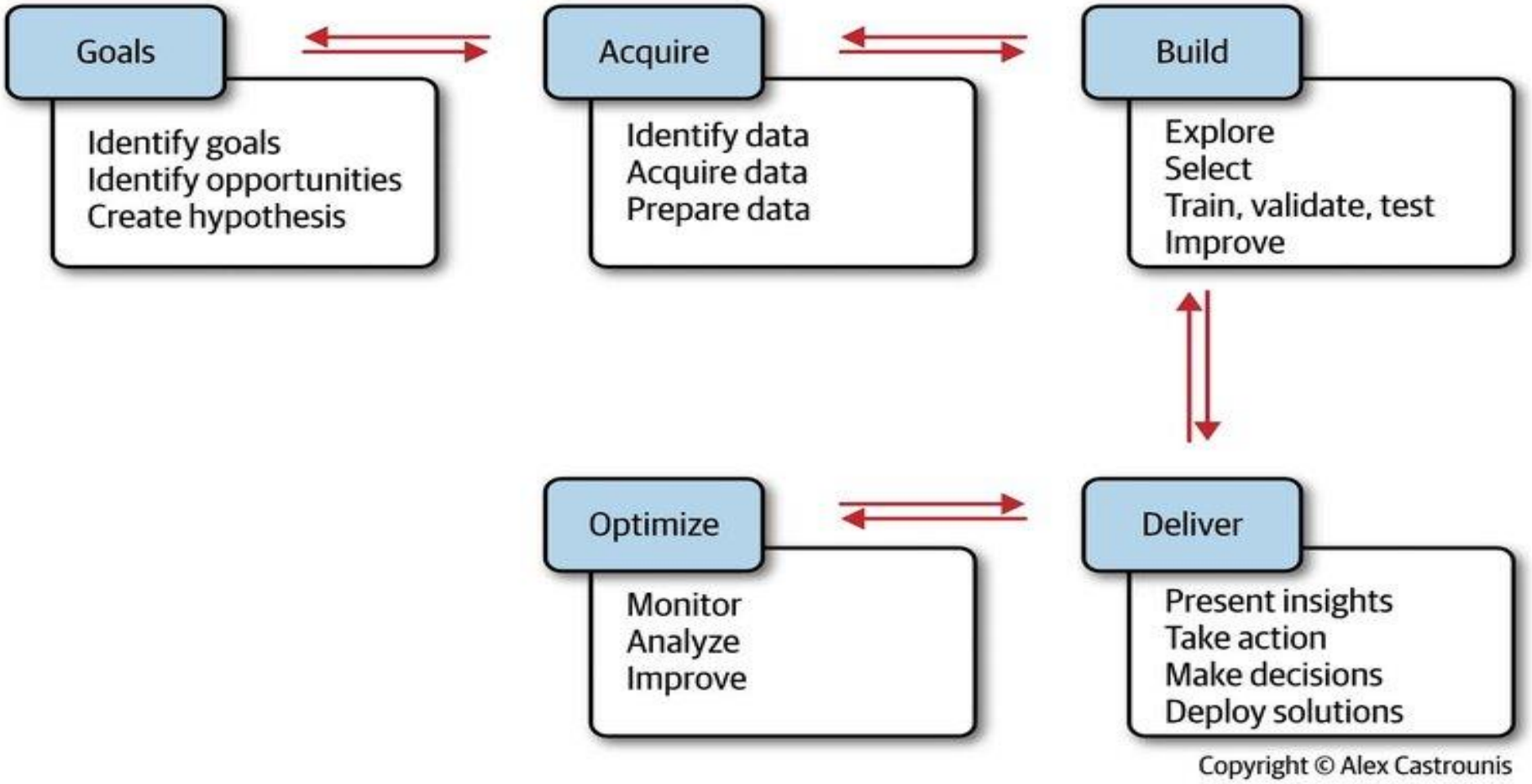


- A data scientist is a person who should be able to leverage existing data sources, and create new ones as needed in order to extract meaningful information and actionable insights.
- A data scientist does this through business domain expertise, effective communication and results interpretation, and utilization of any and all relevant statistical techniques, programming languages, software packages and libraries, and data infrastructure.
- The insights that data scientists uncover should be used to drive business decisions and take actions intended to achieve business goals.

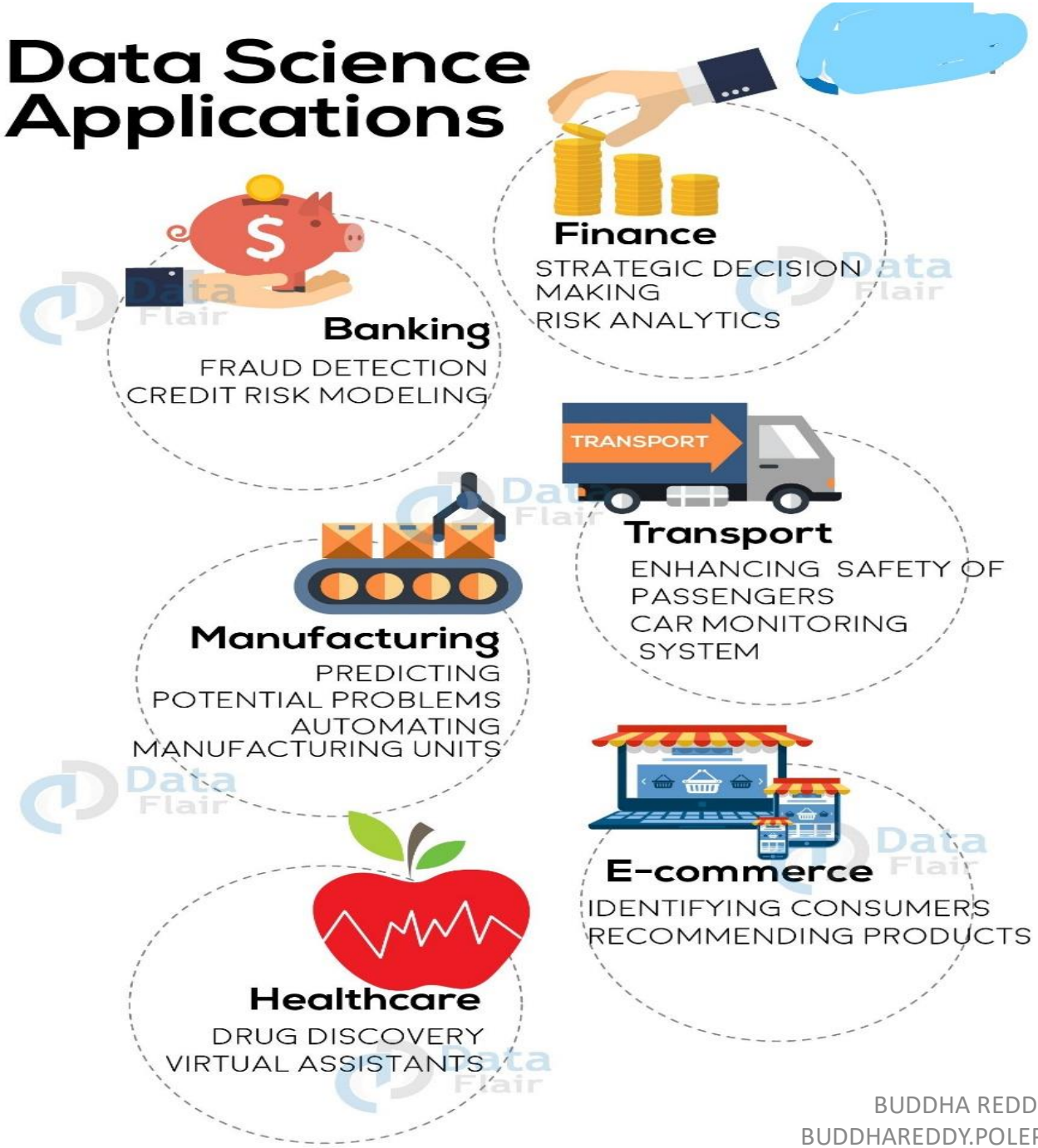
# Data Science Goals and Deliverables

- Prediction (predict a value based on inputs)
- Classification (e.g., spam or not spam)
- Recommendations (e.g., Amazon and Netflix recommendations)
- Pattern detection and grouping (e.g., classification without known classes)
- Anomaly detection (e.g., fraud detection)
- Recognition (image, text, audio, video, facial, …)
- Actionable insights (via dashboards, reports, visualizations, …)
- Automated processes and decision-making (e.g., credit card approval)
- Scoring and ranking (e.g., FICO score)
- Segmentation (e.g., demographic-based marketing)
- Optimization (e.g., risk management)
- Forecasts (e.g., sales and revenue)

# The Data Science Process



# Sectors used




- Retail
- Medicine
- Banking
- Finance
- Construction
- Transportation
- Communication media and entertainment
- Education
- Manufacturing and Natural Resources
- Government
- Energy and utilities
- Gaming etc.

# Healthcare & Pharmaceutical

- if you decide to enter it, ensure that you ready to work around data that could be related to people struggling with their lives.
- So, working in this field is a humanitarian thing to do and it is needless to say
- you have to cautious while designing your data science strategy to offer the most useful conclusions to data problems.

# Telecommunications Sector

- Mind Commerce anticipates big data and data science industry to escort the telecom sector to expand at a compound annual growth rate of 50 percent, with the annual revenue reaching to $5.4 Billion by the end of 2019.
- Data storage costs have gone way down and computer processing power is going skywards due to easily available analytics software.
- Hence, the job of a data analyst has become a bit easier.
- Data collected through customer behaviors, such as voice, video choices, SMS, social media activities, demographic is enabling telecom companies to offer customized services as well as products.

# Internet Industry

- The internet industry is gaining strength as we speak as a result of the data surge caused by sophisticated technology, big data along with cloud computing.
- There is an inexplicable amount of data in the hands of data scientists which they are using to design personalized recommendation, undertake sentiment analysis, video analysis, etc.
- Hence, the internet industry, e-commerce, and social networks are thriving to an unimaginable level with billions and billions of people using the internet, posting pictures and videos on social media and making Google searches every second of the day.

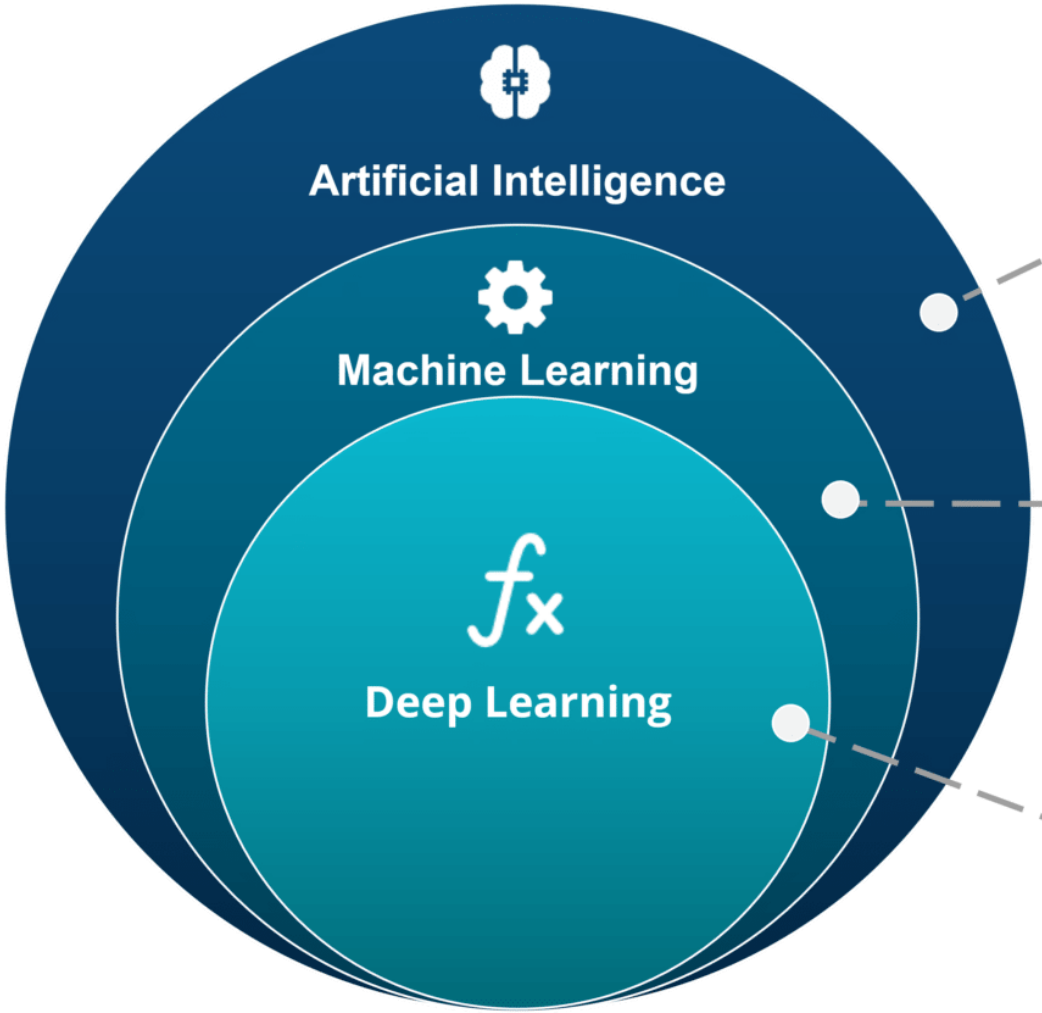
# Energy Sector

- Data science and big data are exhibiting their transformational powers to reshape the Energy vertical.
- Data analysts are proving to be highly useful for discovering unconventional energy sources, reducing costs and saving money on exploration and drilling, increasing efficacy, avoiding power outages, enhancing productivity, and so on.
- The data science industry is also curbing the chances of accidents by offering better repairs.

.

# Automotive Industry

- Data science professionals have a huge role to play in the automotive industry.
- Artificial Intelligence, machine learning, data science are the main technologies enabling this sector to combat its various challenges and refurbish itself.
- Working in this sector will give you a great chance to widen the range of your skills by creating products with optimization and automatic learning




## ARTIFICIAL INTELLIGENCE

A technique which enables machines to mimic human behaviour

## MACHINE LEARNING

Subset of AI technique which use statistical methods to enable machines to improve with experience

## DEEP LEARNING

Subset of ML which make the computation of multi-layer neural network feasible


WHAT IS
A.I.?

- The term **artificial intelligence** **has become more popular these days**

- **Why its become more popular these days?**

**Tremendous increase in data volumes, advanced algorithms, and improvements in computing power and storage.**

- **What is Artificial Intelligence?**

**Allows the machines to act like humans by replicating their behavior and nature.**

# Are we using any AI in our day to day life ?

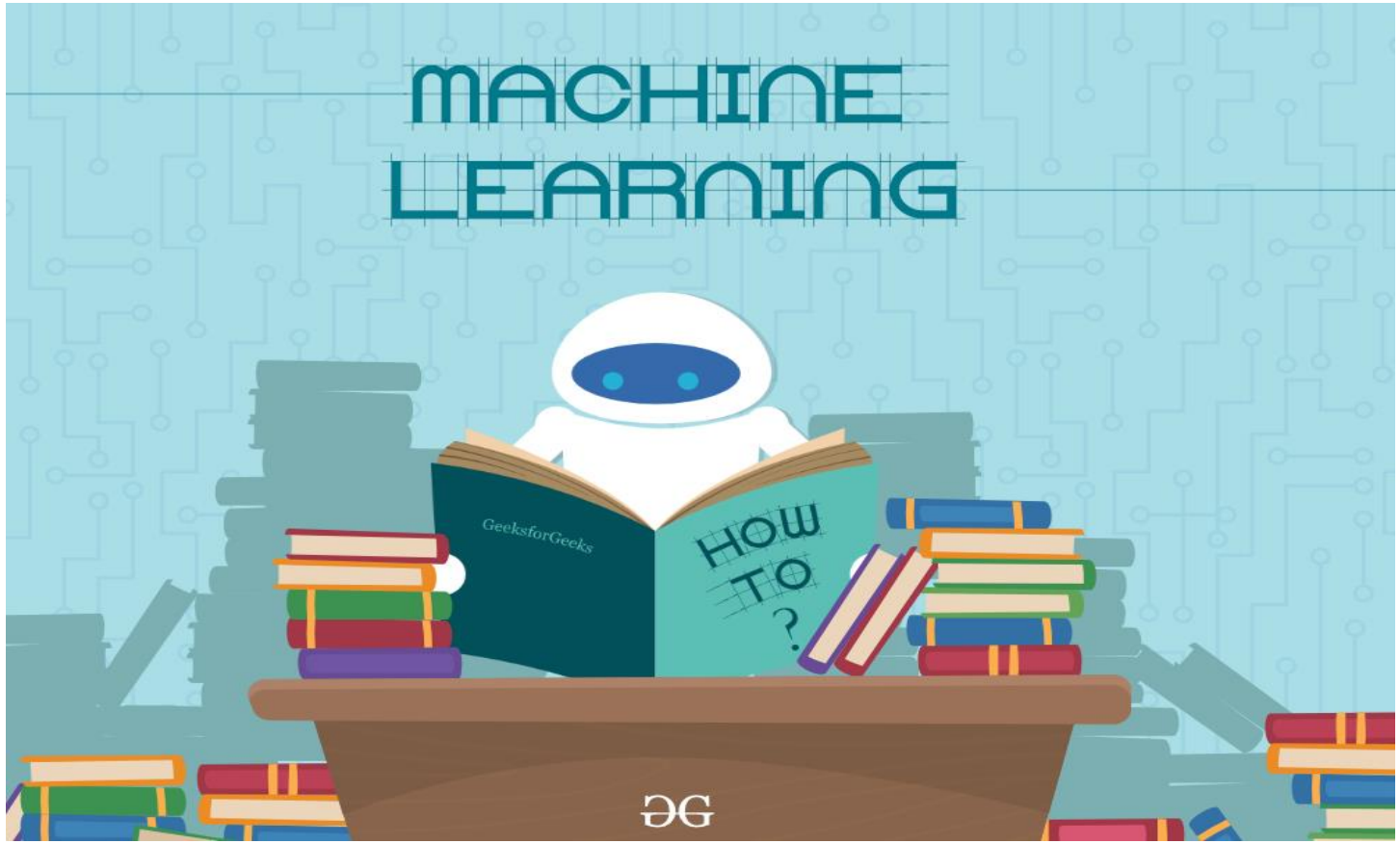
- Apple's Siri
- Alexa
- Google virtual assistant
- The chess-playing computer
- Tesla's  self-driving car and many more.
- These examples are based on Deep Learning and Natural Language Processing.

# What is Artificial Intelligence (AI)?

- Artificial Intelligence (AI) refers to the **simulation of human intelligence processes by machines, including learning, reasoning and self-correction**.
- AI allows machines to learn from their experience.
- The machines **adjust their responses** to **perform human activities** by **processing data and recognizing patterns in them.**
- AI can be of two types**: weak AI and strong AI.**
- **Weak Artificial Intelligence** refers to an AI system developed for a specific task.
- **Strong Artificial Intelligence** is an AI system with generalized human cognitive skills.

# Application

- To solve customer service issues,
- Inform people about the latest news along with giving them live traffic updates
- Weather  forecast.


MACHINE
LEARNING
GeeksforGeeks
HOW
TO
?

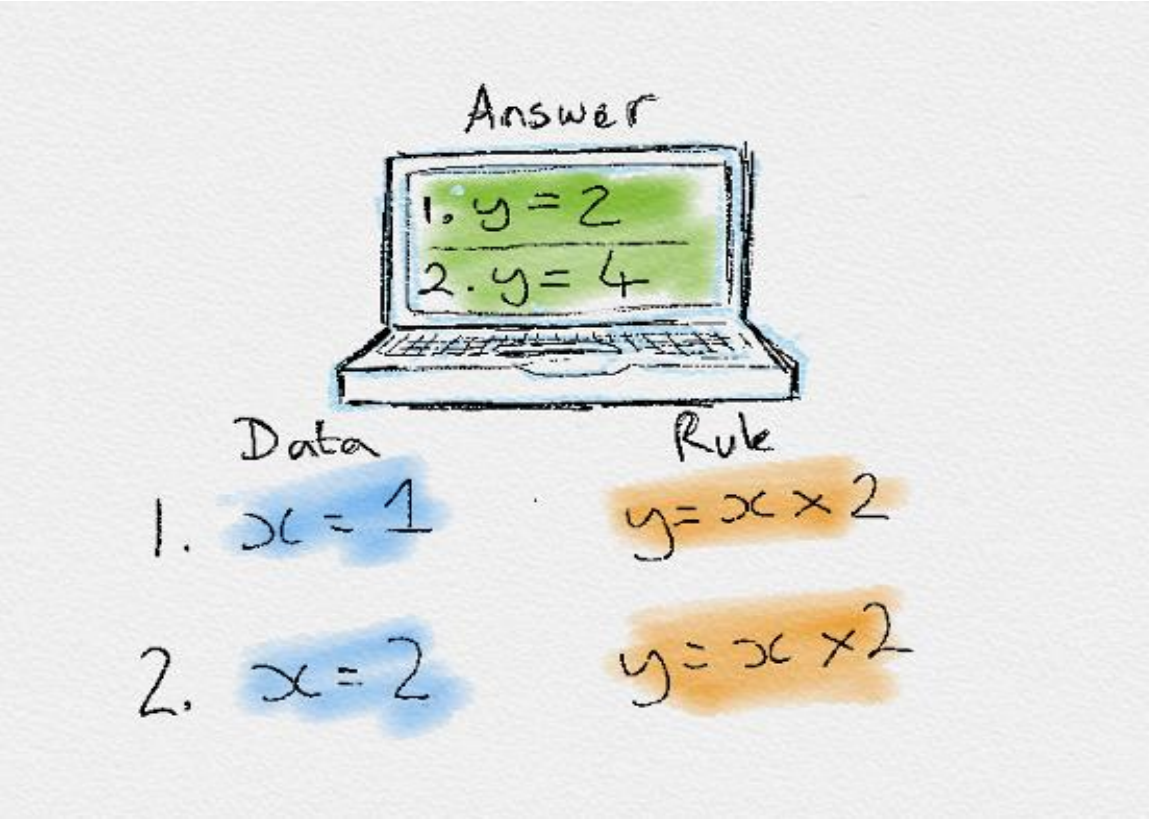
# What is Machine Learning (ML)?

- ML enables **machines to learn by themselves through the use of given information.**
- It allows the **machines to learns from the past experiences (input data) and make predictions based on its experience(data)“**
- The performance of such a system should be at least human level.

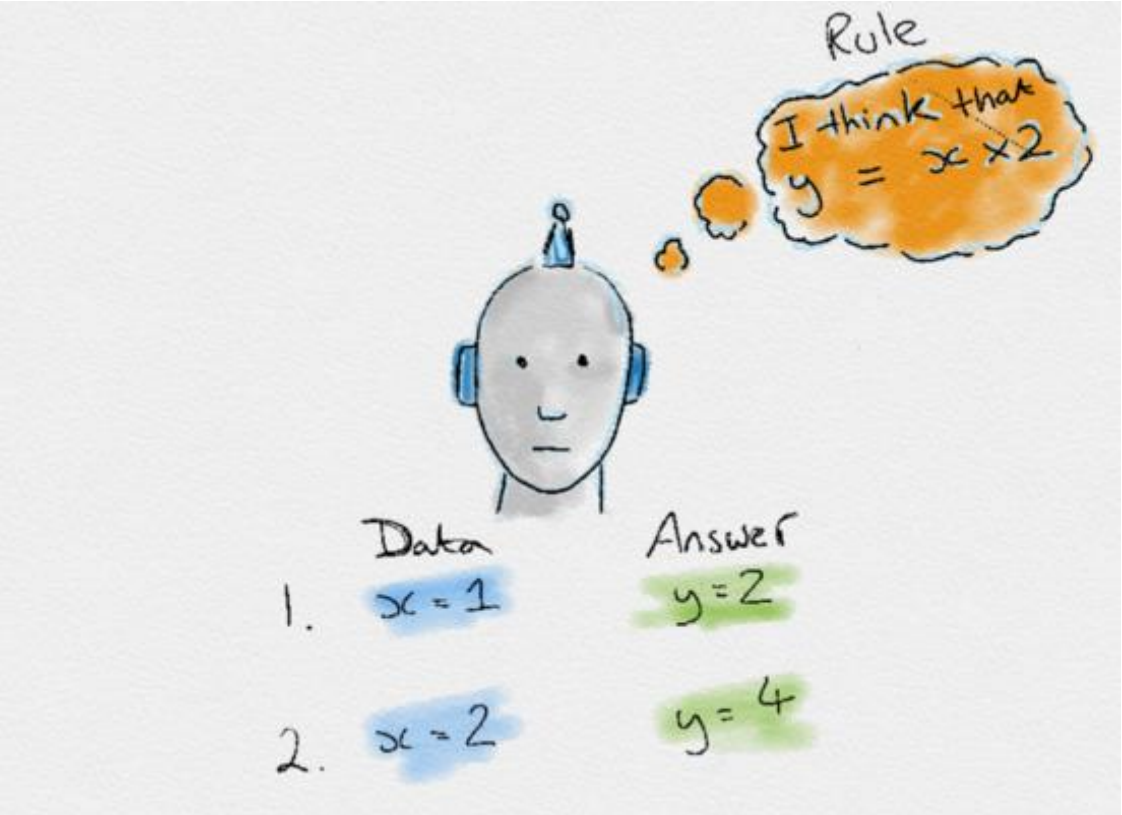
**Application:**

Machine Learning is often used to power recommendation engines that provide suggestions based on past customers’ behaviors

## Traditional Programming




## Machine Learning



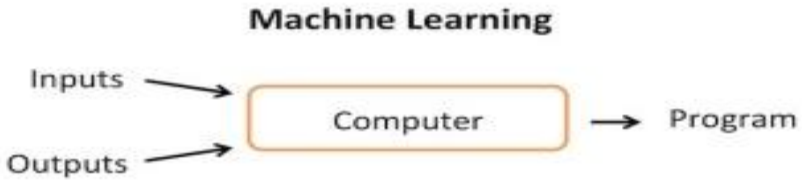

Software engineering combined human created *rules* with *data* to **create answers to a problem**.

Machine learning uses *data* and *answers* to **discover the rules behind a problem**.

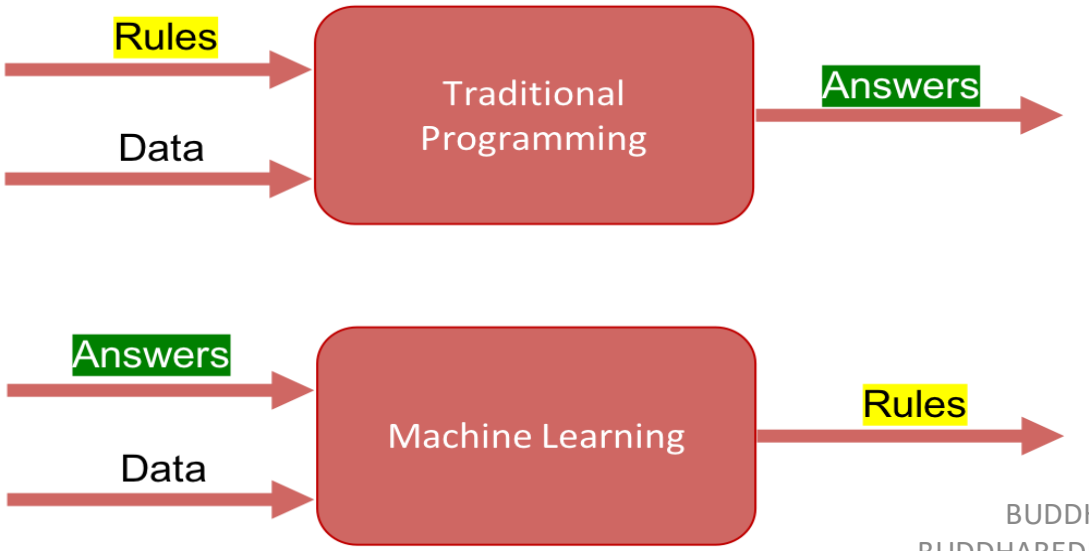


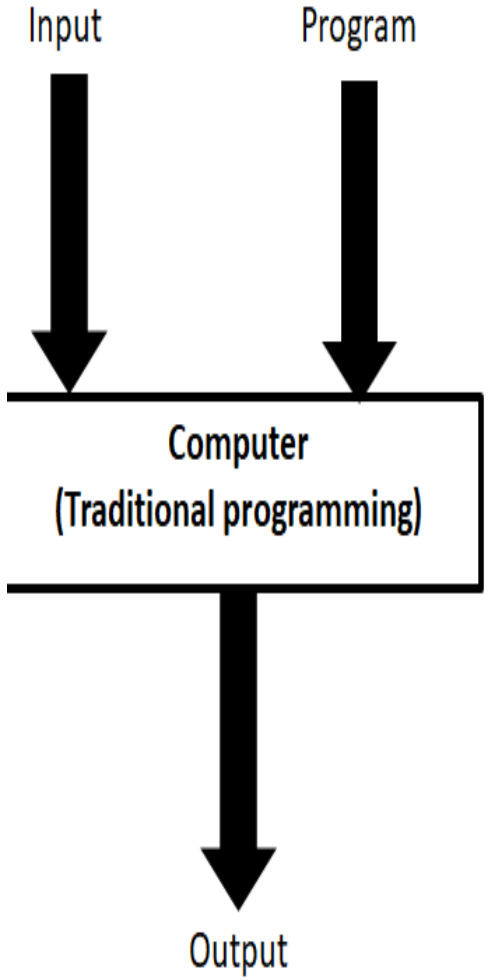
------(Chollet, 2017)

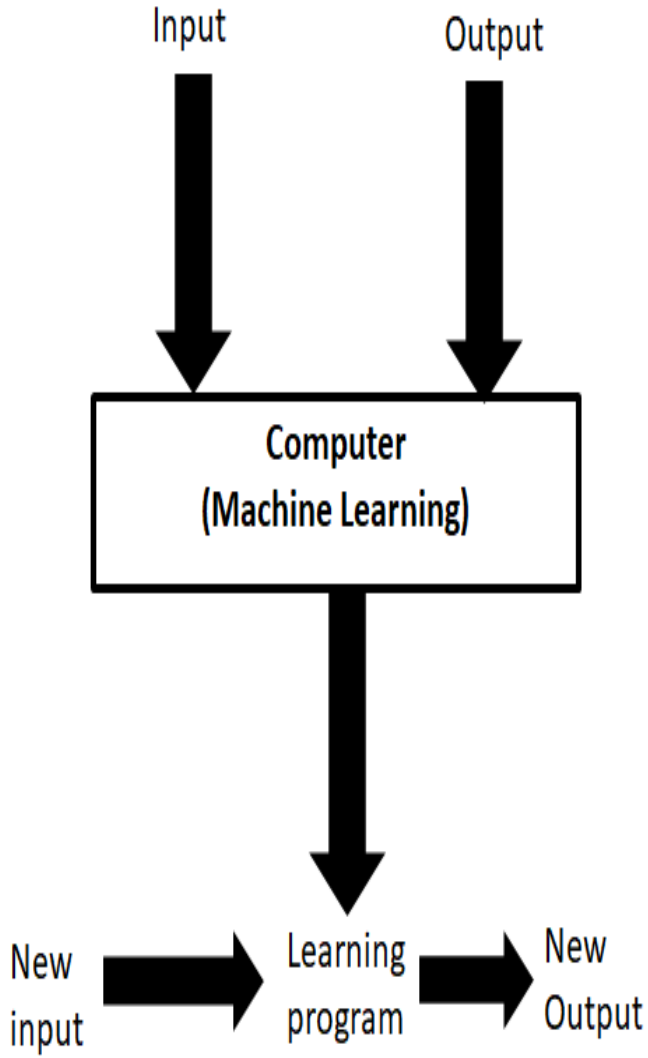

*Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed*
– Arthur Samuel (1959)



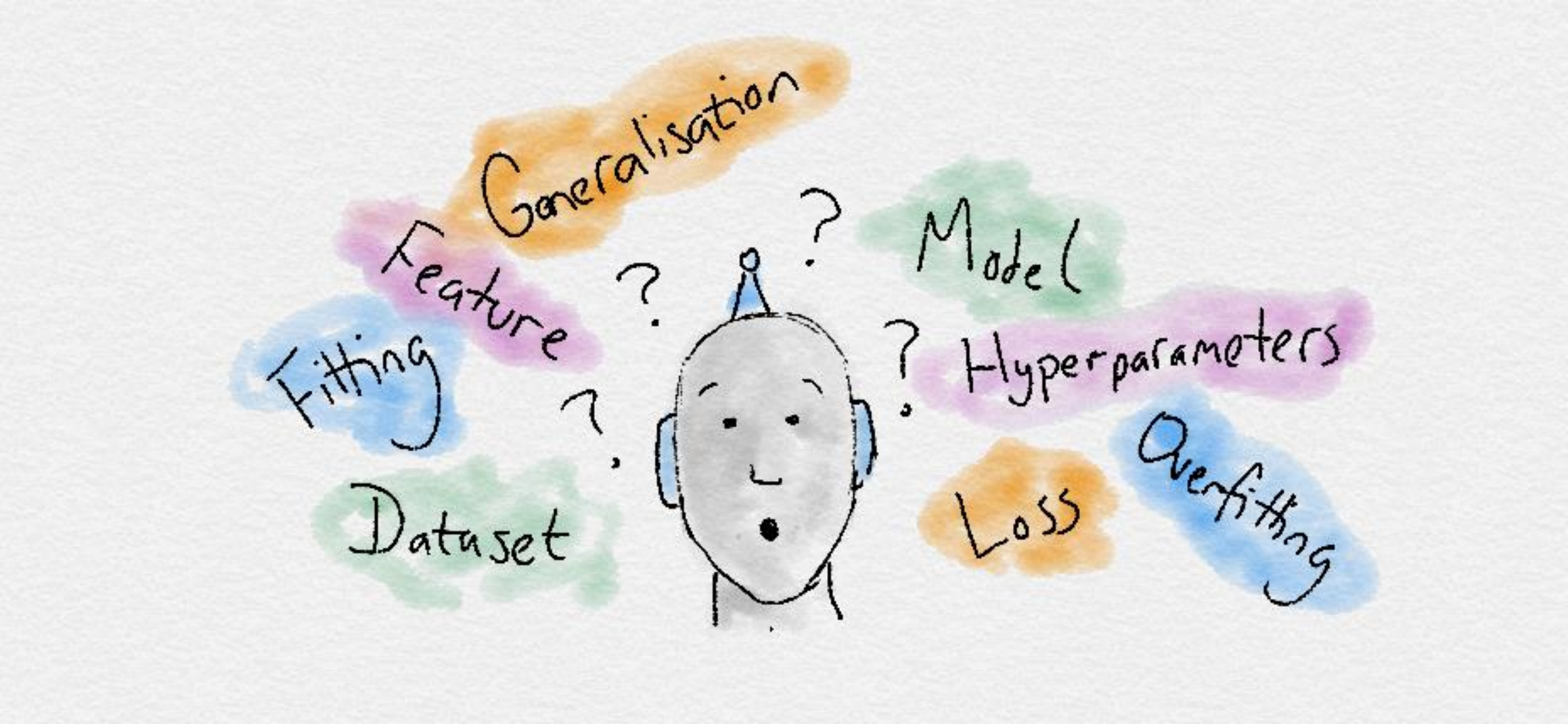

Sebastian Raschka, 2016



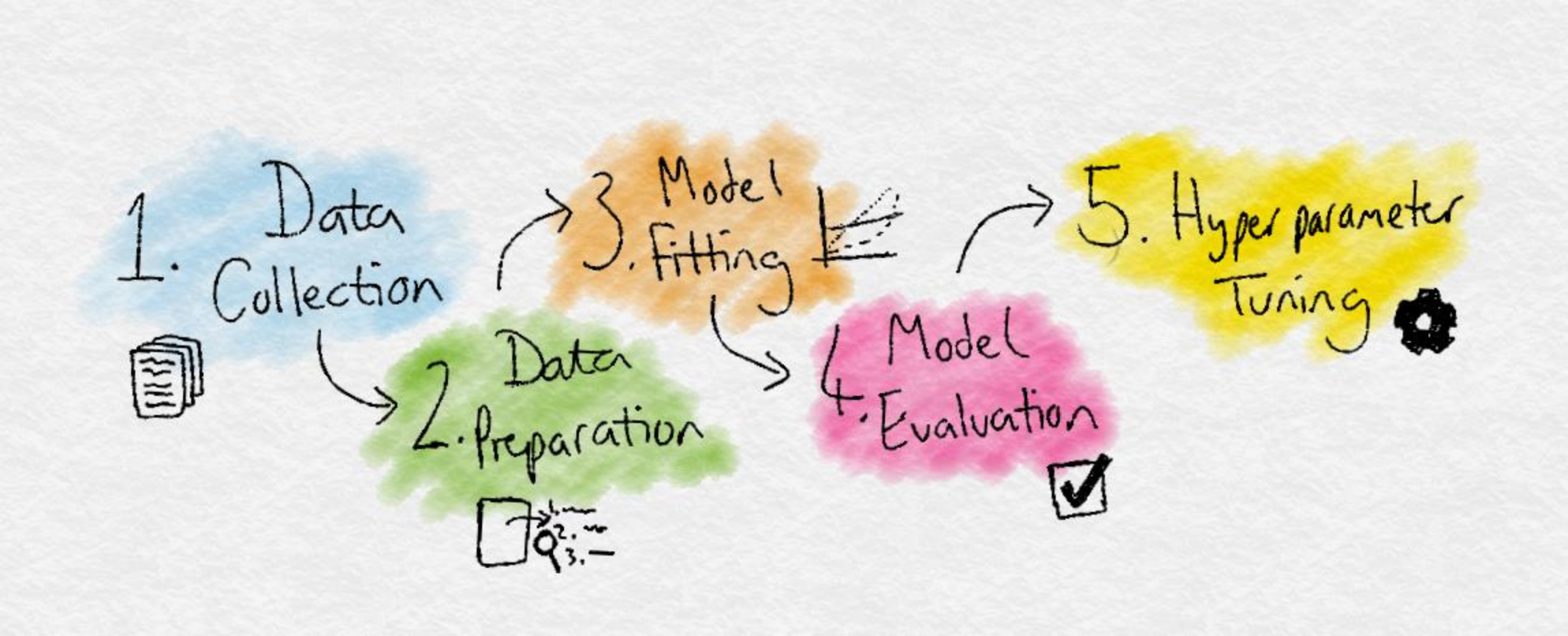



# Terminology



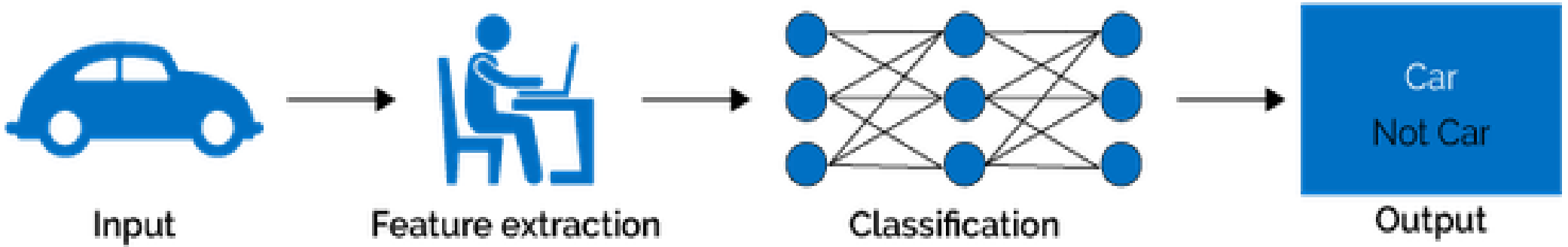
- **Dataset**: contain features important to solving the problem.
- **Features**: Column names.
- **Model**: The representation (internal model) of a a Machine Learning algorithm has learnt. It learns this from the data it is shown during training.
- The model is the output you get after training an algorithm.
- For example, a decision tree algorithm would be trained and produce a decision tree model.
- **Hyperparameter** is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.

# Process

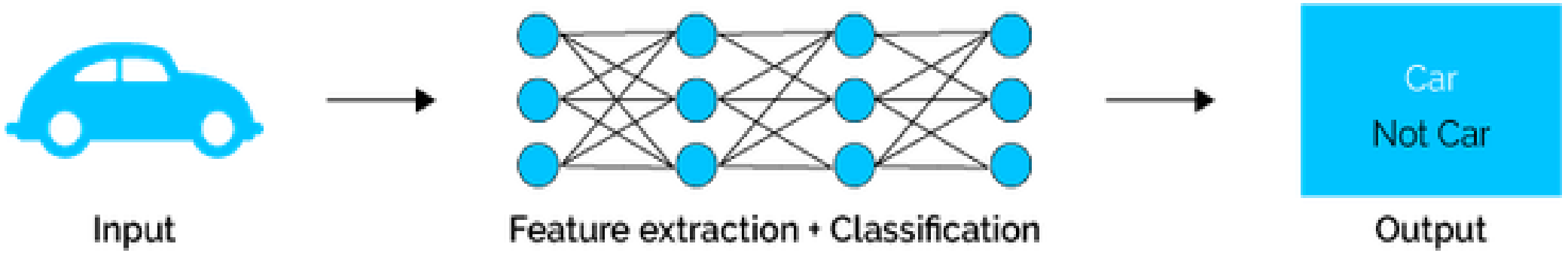

1. Data Collection
2. Data Preparation
3. Model Fitting
4. Model Evaluation
5. Hyper parameter Tuning

- **Data Collection:** Collect the data that the algorithm will learn from.
- **Data Preparation:** Extracting important features & performing dimensionality reduction.
- **Training:** Also known as the fitting stage, this is where the Machine Learning algorithm actually learns by showing it the data that has been collected and prepared.
- **Evaluation:** Test the model to see how well it performs.
- **Tuning:** Fine tune the model to maximize it's performance

# Machine Learning




# Deep Learning



# What is Deep Learning (DL)?

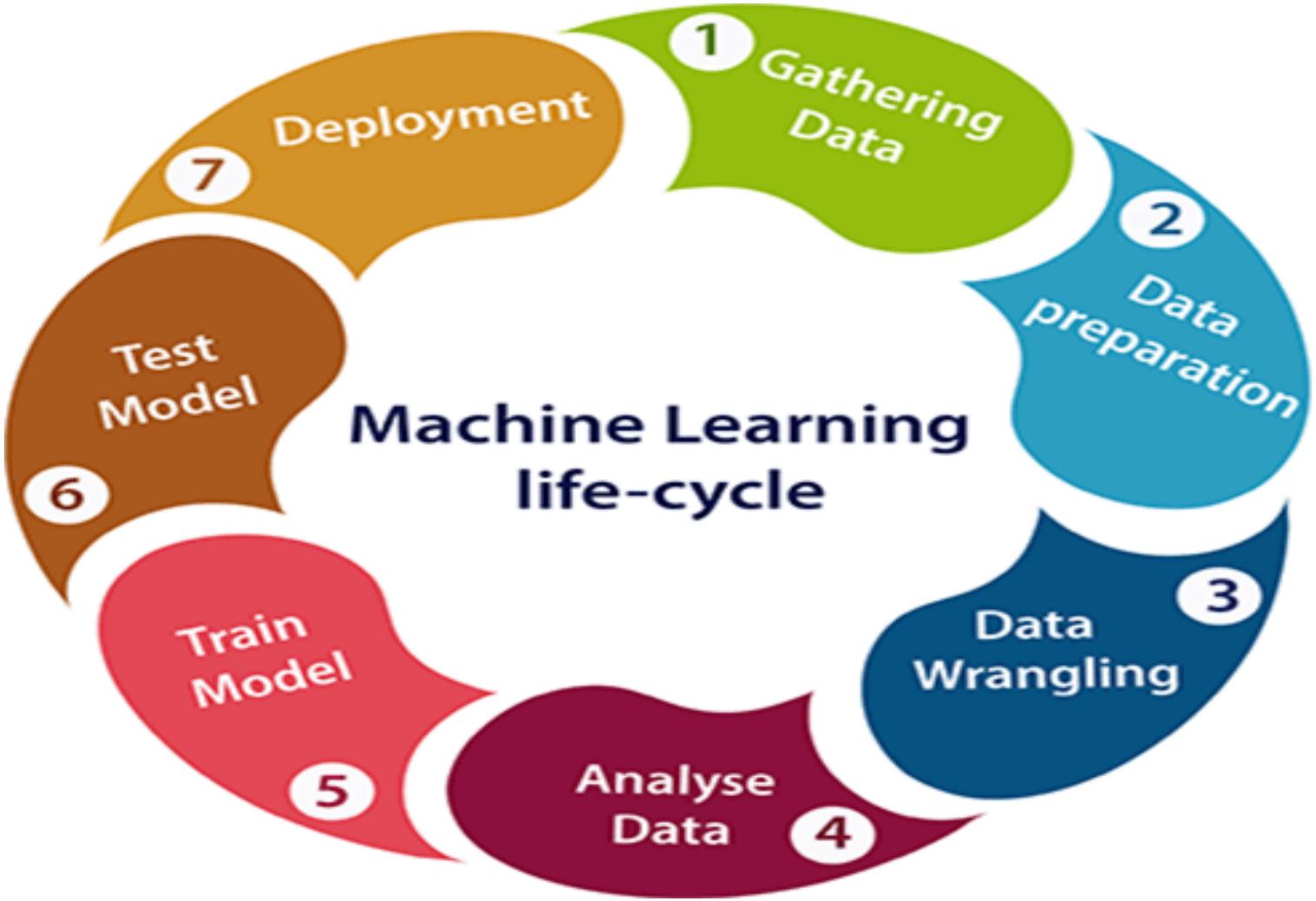
- "Deep learning is a particular kind of machine learning that **achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts or abstraction**".
- The term refers to a particular approach used for **creating and training neural networks that are considered highly promising decision-making nodes.**
- **DL is inspired by the functionality of our brain cells called ANN.**

## Application

Deep Learning is used to develop highly automated systems such as self-driving cars. Through their sensors and onboard analytics, these cars can reorganize obstacle and facilitate situational awareness

# Machine learning Life cycle

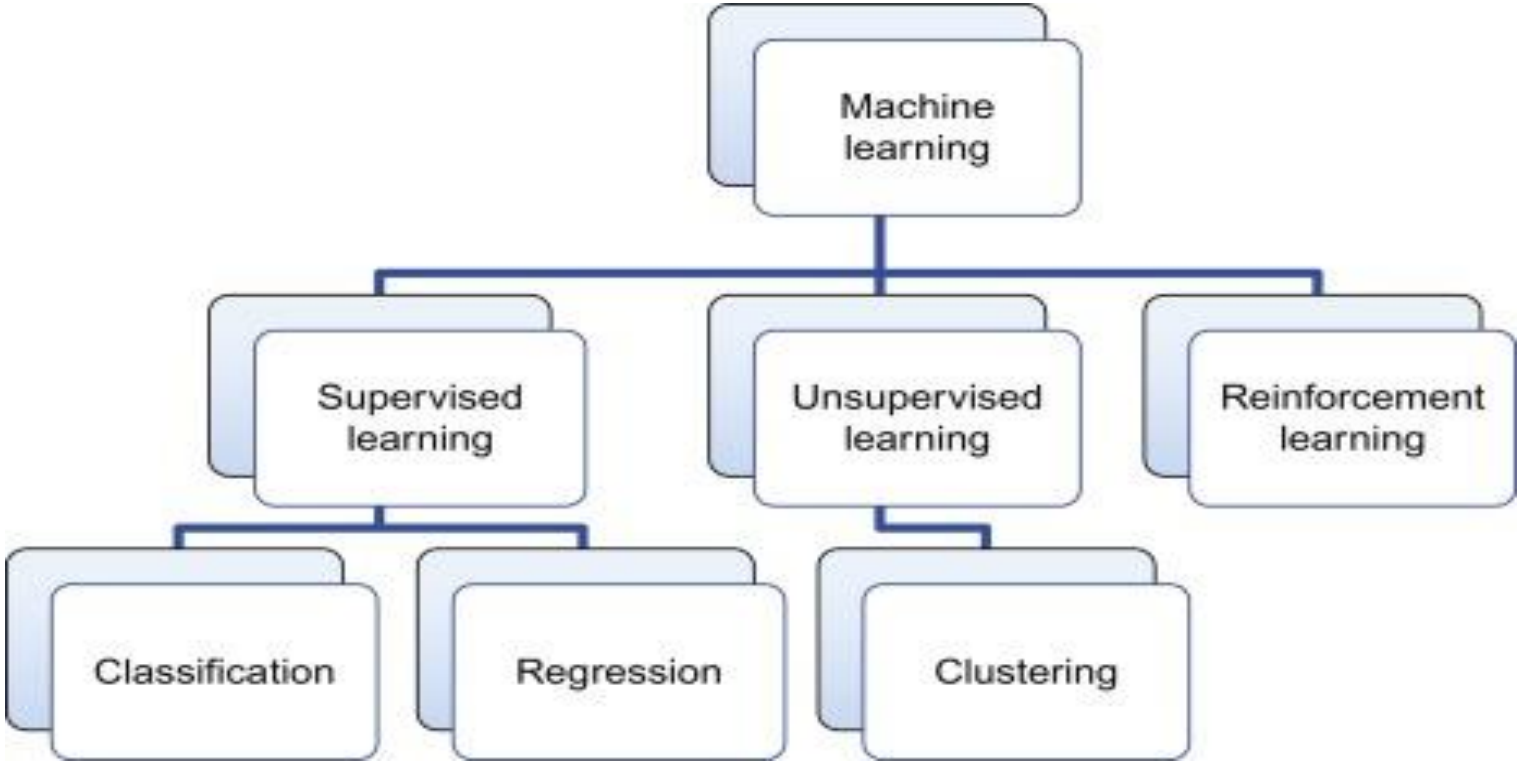
- Data wrangling is the process of cleaning and converting raw data into a useable format.
- It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step.




- Cleaning of data is required to address the quality issues.
- In real-world applications, collected data may have various issues, including:
  - **Missing Values**
  - **Duplicate data**
  - **Invalid data**

# Machine learning algorithms categorized into

- **Supervised**
- **Unsupervised**
- **Semi Supervised**

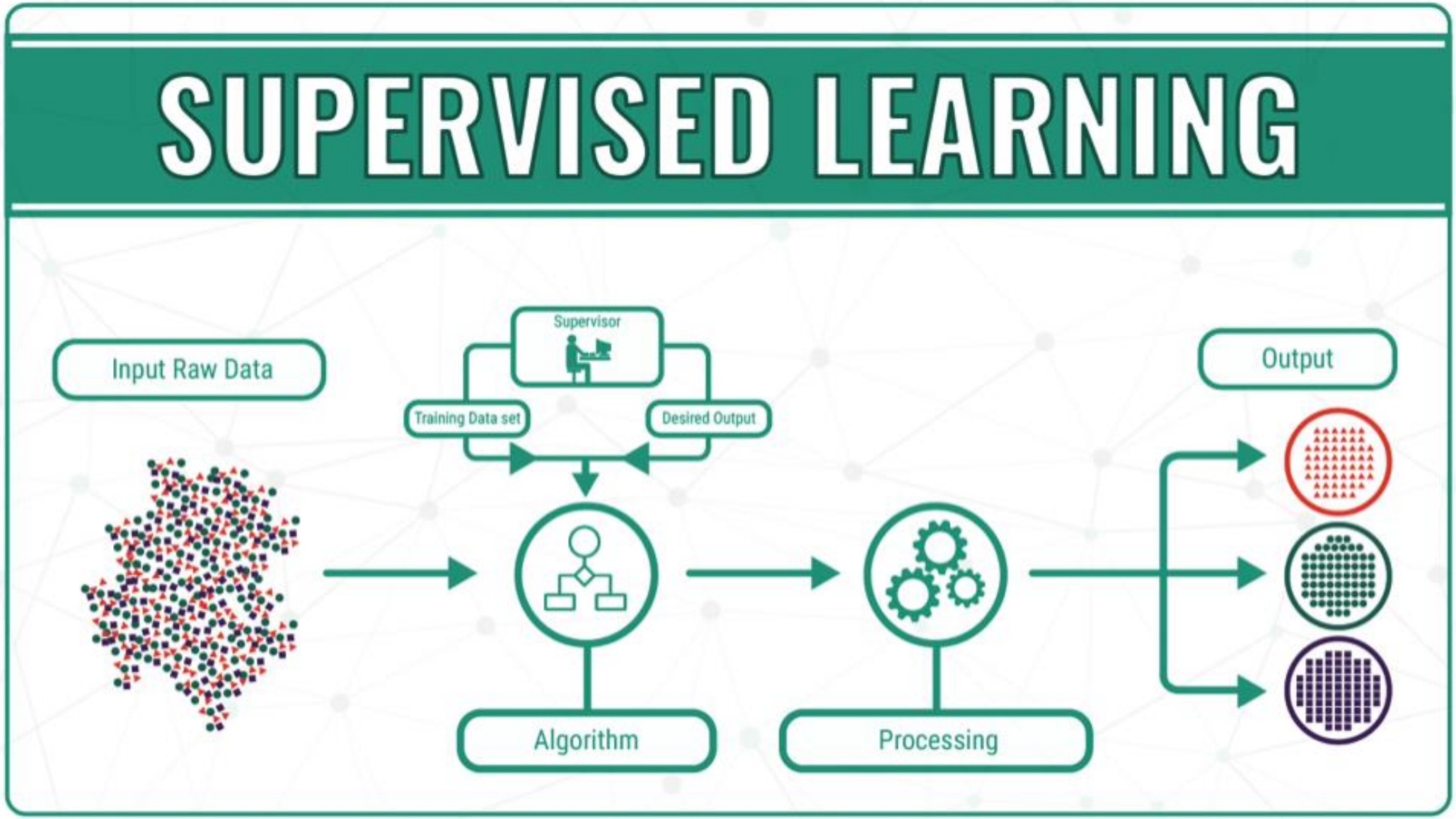


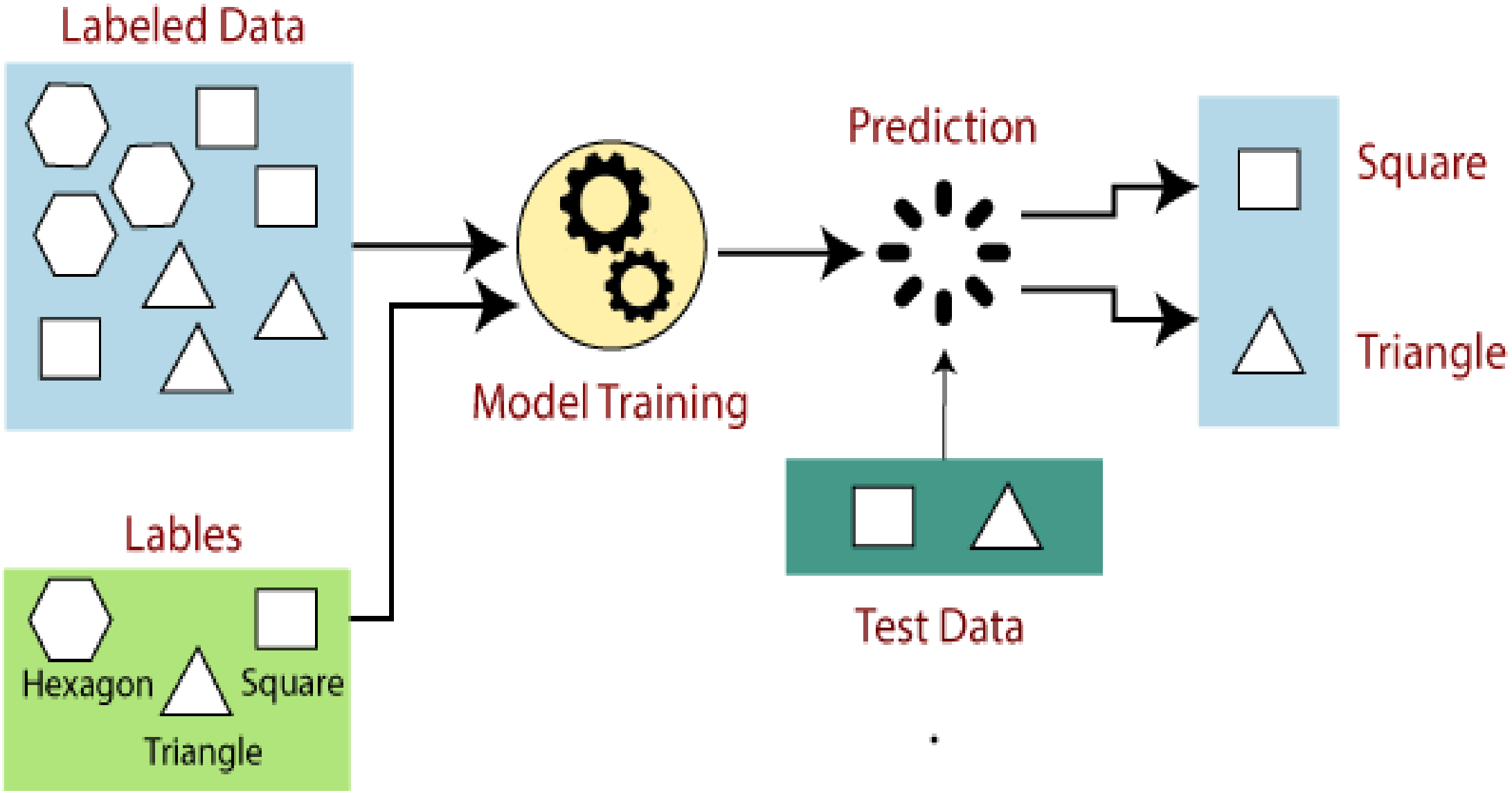
# Analogy

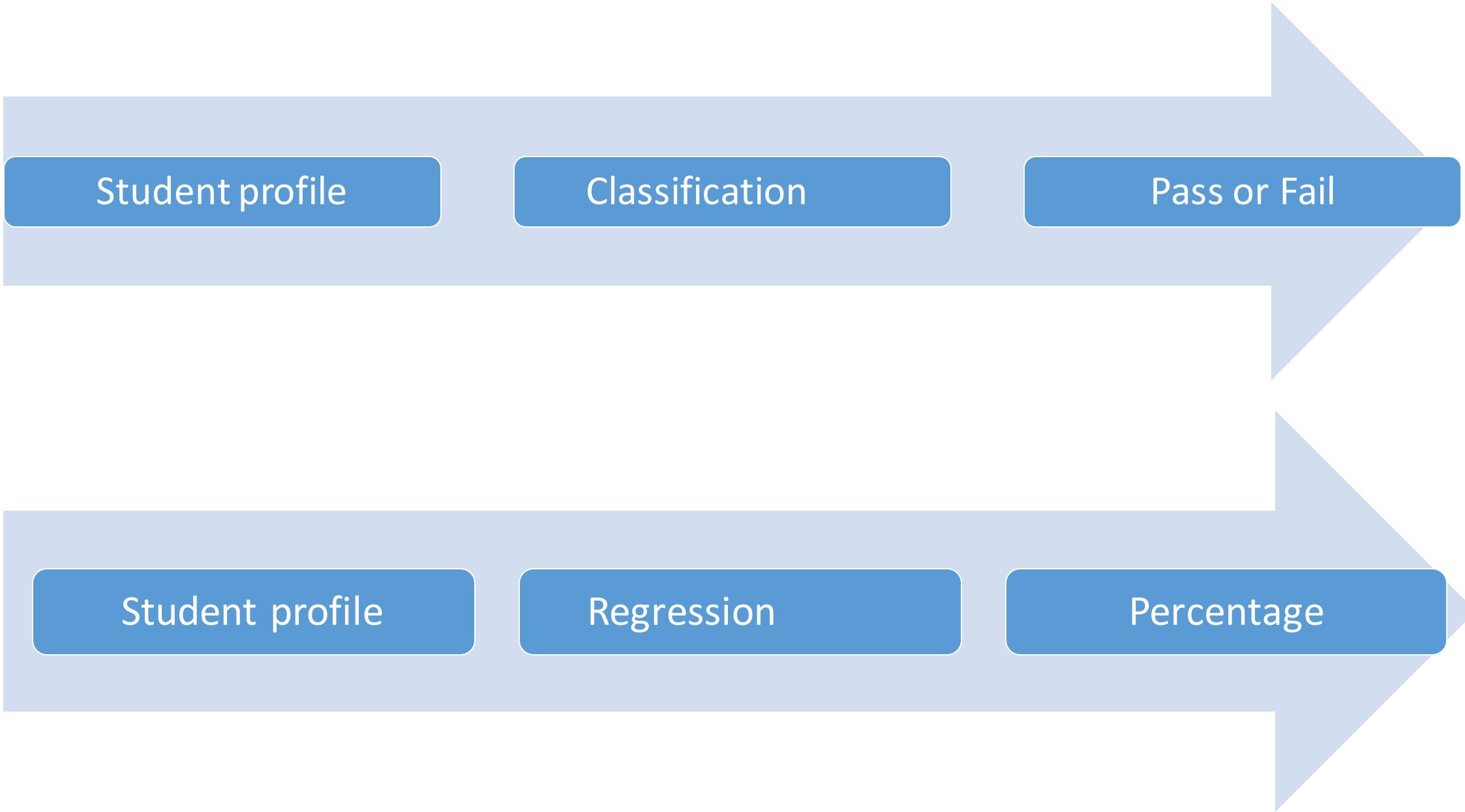
- The **inputs** be **weather forecast**, and **outputs** be the **visitors to the beach**.
- The goal would be to **learn the mapping that describes the relationship** between **temperature and number of beach visitors**.
- **labelled** data is provided of past **input and output pairs** during the learning process to teach the model how it should behave.
- **New inputs** can then be fed in of forecast temperature and the Machine learning algorithm will then **output a future prediction** for the number of visitors.

# Supervised Machine Learning

- Supervised learning is where you have **input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output**.
- The goal is to **approximate the mapping function** so well that **when you have new input data (x) that you can predict the output variables (Y) for that data.**
- Supervised learning problems grouped into **Classification & regression problems.**
- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” , “ yes” or “No”, “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” ,”salary”, ”cost” , “weight”.

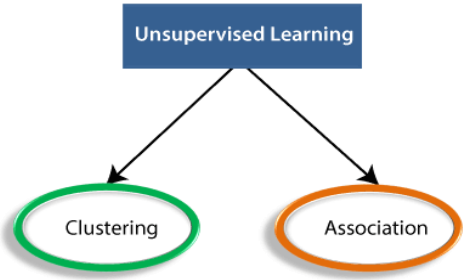

SUPERVISED LEARNING
Input Raw Data
Supervisor
Training Data set
Desired Output
Algorithm
Processing
Output


Labeled Data
Lables
Hexagon
Square
Triangle
Model Training
Prediction
Square
Triangle
Test Data

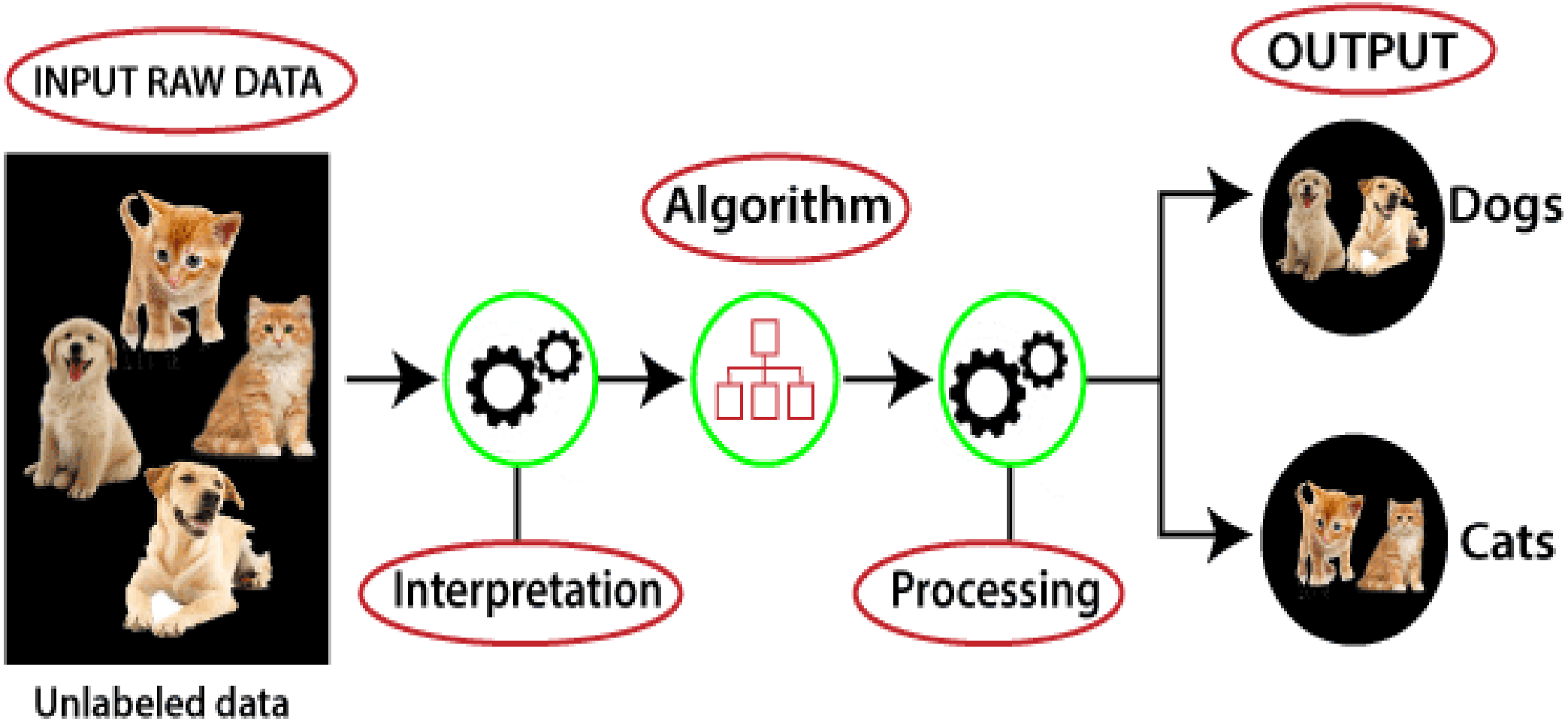

Student profile
Classification
Pass or Fail
Student profile
Regression
Percentage

# Unsupervised Machine Learning

- Here you only **have input data (X) and no corresponding output variables**.
- The goal for unsupervised learning is to **model the underlying structure or distribution in the data in order to learn more about the data**.
- In this we discover and **present the interesting structure in the data.**

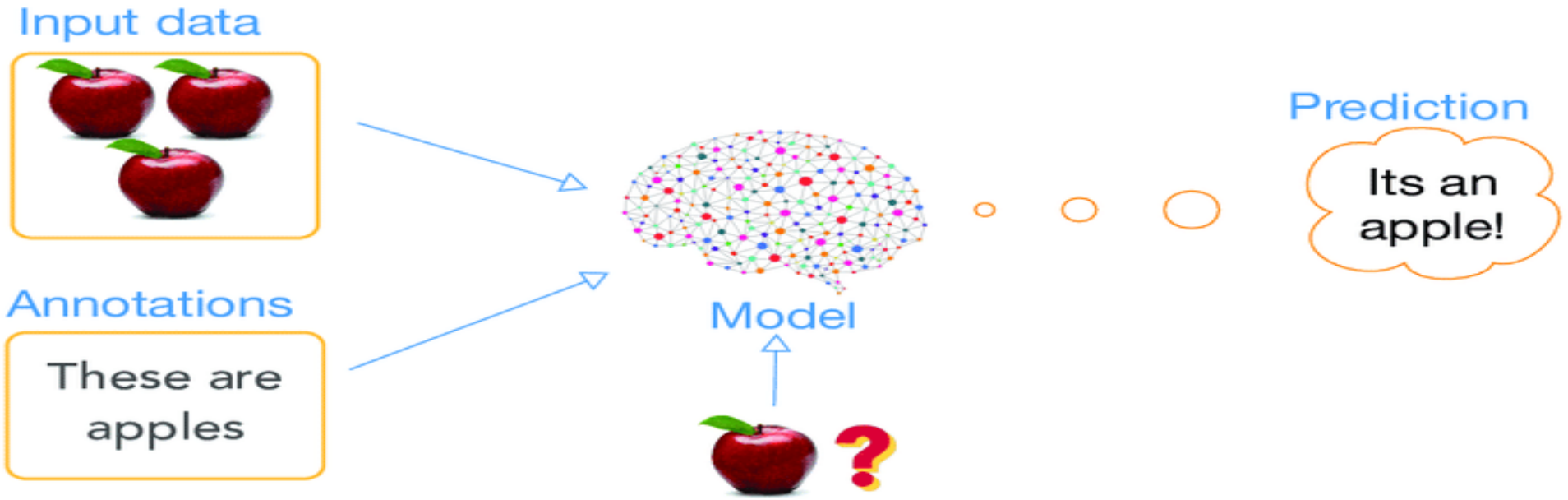



- Unsupervised learning problems grouped into **clustering** and **association problems.**
- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

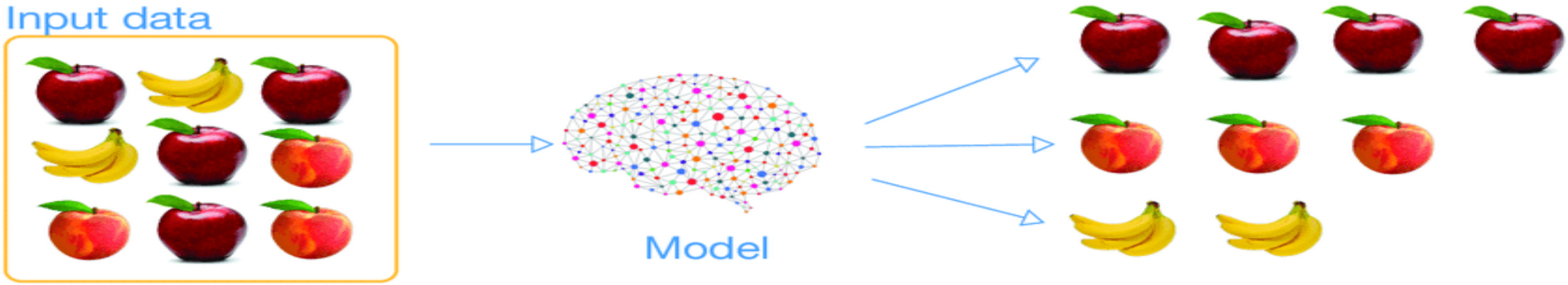

INPUT RAW DATA
Unlabeled data
Interpretation
Algorithm
Processing
OUTPUT
Dogs
Cats

- **Clustering:** Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database.
- It determines the set of items that occurs together in the dataset.
- Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item.
- A typical example of Association rule is Market Basket Analysis.

supervised learning




unsupervised learning



# Semi-Supervised Machine Learning

- Problems **where you have a large amount of input data (X) and only some of the data is labeled (Y)** are called semi-supervised learning problems.
- A good example is a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.
- This is because it can be expensive or time-consuming to label data as it may require access to domain experts.
- **unlabeled data is cheap and easy to collect and store.**

# Types of ML algorithms in Supervised Learning

## Classification

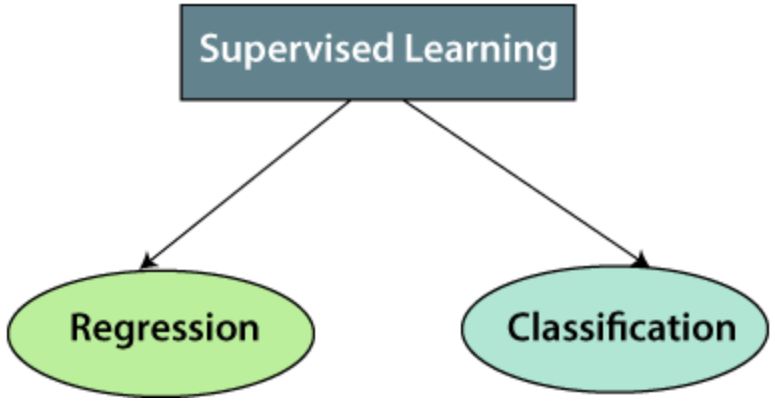
1. K-Nearest Neighbor
2. Naive Bayesian
3. Decision tree
4. Random forest
5. Logistic Regression
6. Neural network
7. Support Vector Machine

## Regression

1. Simple linear
2. Multilinear
3. K-Nearest Neighbor
4. Decision tree
5. Random forest
6. Polynomial
7. Stepwise
8. Regularization: Lasso, Ridge
9. Elastic net

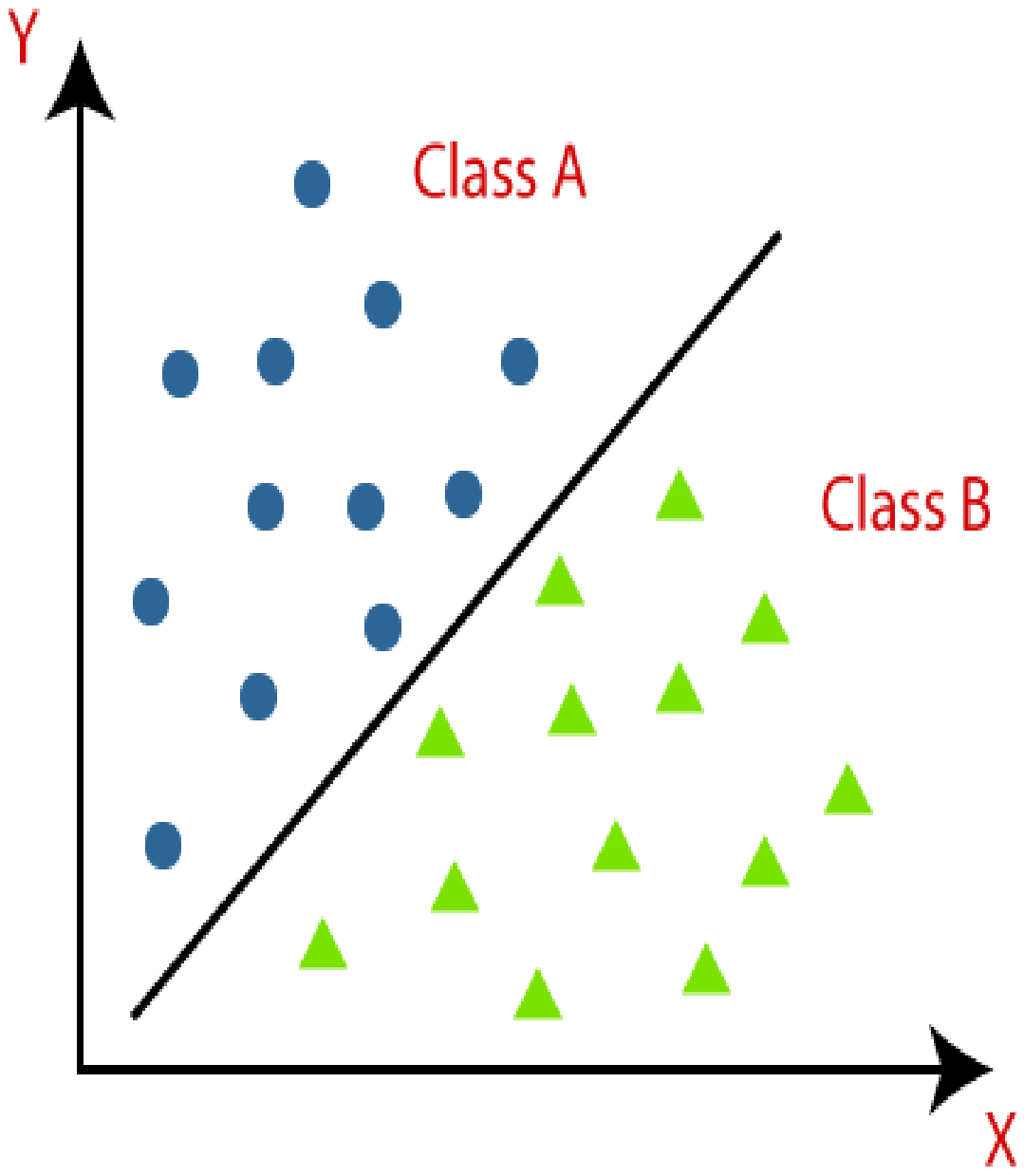
# Supervised Machine Learning Algorithms

- In supervised learning, models are trained using labelled dataset, where the model learns about each type of data.
- Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.



# Classification Algorithm

- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.
- In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups.
- Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc.
- Classes can be called as targets/labels or categories.
- The output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc.
- Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.




- The algorithm which implements the classification on a dataset is known as a classifier.
- There are two types of Classifications:
- **Binary Classifier:** If the classification problem has only two possible outcomes.
- **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes.

  **Example:** Classifications of types of crops, Classification of types of music.

- In the classification problems, there are two types of learners:
- **Lazy Learners:** Lazy Learner firstly stores the training dataset and wait until it receives the test dataset. In Lazy learner case, classification is done on the basis of the most related data stored in the training dataset. It takes less time in training but more time for predictions.

**Example:** K-NN algorithm, Case-based reasoning

- **Eager Learners:** Eager Learners develop a classification model based on a training dataset before receiving a test dataset. Opposite to Lazy learners, Eager learners take less time in training and more time in prediction. **Example:** Decision Trees, Naïve Bayes, ANN.

# Terminology

- **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.
- **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
- **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values.
- An outlier may hamper the result, so it should be avoided.

- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity.
- It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
- **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**.
- And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

# K NEAREST NEIGHBOR

# let’s take a real life example and understand.

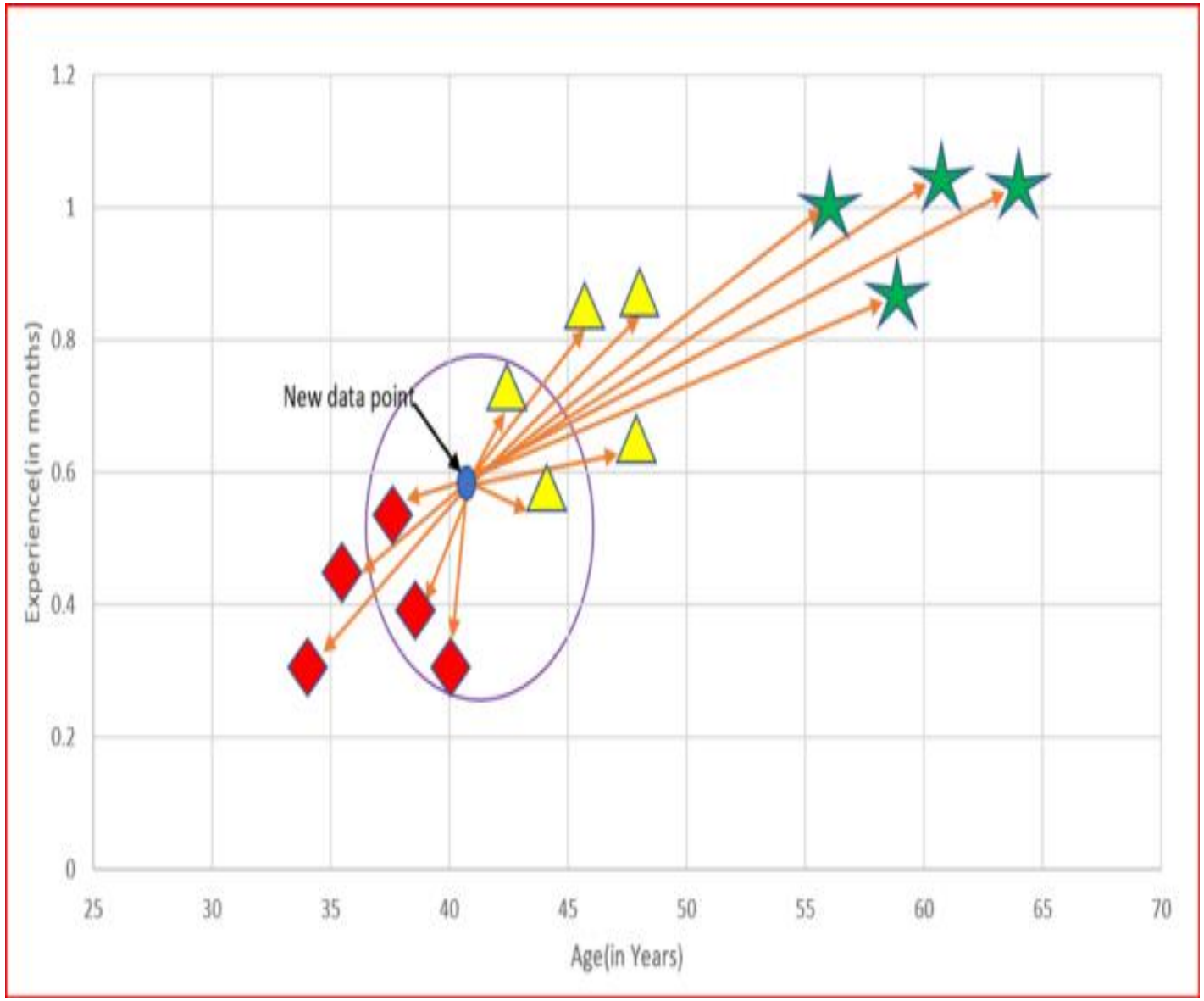
- You moved to a new neighborhood and want to be friends with your neighbors.
- You start to socialize with your neighbors.
- You decide to pick neighbors that match your thinking, interests and hobbies. Here thinking, interest and hobby are features.
- You decide your neighborhood friend circle based on interest, hobby and thinking similarity.
- This is analogous to how KNN works

# What is *K- Nearest neighbors (*KNN) Algorithm?

- KNN is a simple algorithm which uses the entire dataset in its training phase.
- Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction.
- Used for both **Classification and Regression**
- Uses **feature similarity** to predict the cluster that the new point will fall into.
- To determine which of the K instances in the training dataset are most similar to a new input a **distance measure** is used.
- For real-valued input variables, the most popular distance measure is **Euclidean distance.**

# KNN named as

- **Instance based learning:** Instead of performing explicit generalization, we are comparing new problem instances with the instances in training data which is stored in memory (where each training instance is a case from the problem domain).
- **Lazy Learning:** No learning of the model is required and all of the work happens at the time a prediction is requested
- **Non Parametric:** KNN makes no assumptions about the functional form of problem being solved

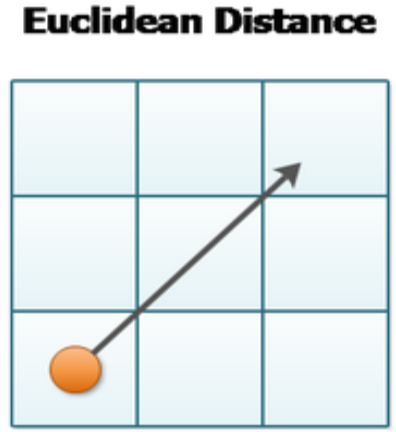



**k** in **kNN** algorithm represents the number of nearest neighbor points which are voting for the new test data's class.

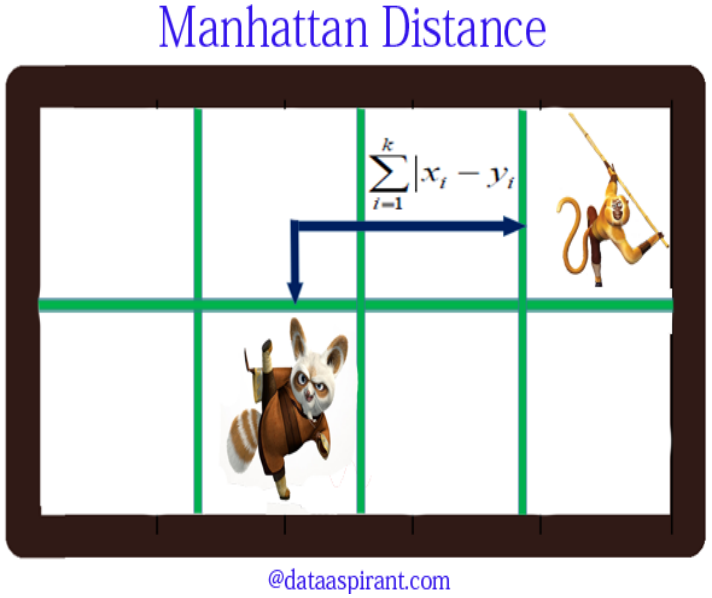
## *How is the distance calculated?*

Distance can be calculated using

- **Euclidean distance**
- **Manhattan distance**
- **Hamming Distance**
- **Minkowski Distance**




$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

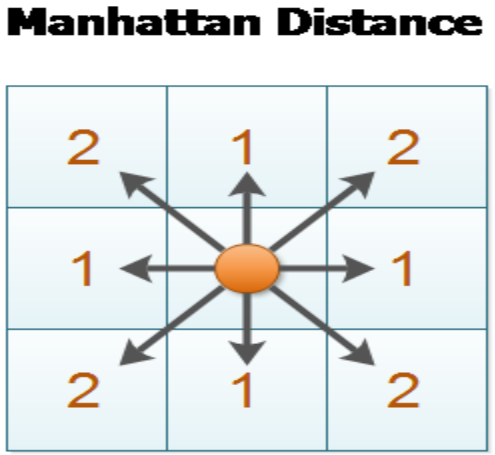



- **Euclidean distance** is the square root of the sum of squared distance between two points. It is also known as L2 norm.

Euclidean Distance between $(x_1, y_1)$ and$(x_2, y_2)$ = $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

- **Manhattan distance** is the sum of the absolute values of the differences between two points

Manhattan Distance between $(x_1, y_1)$ and$(x_2, y_2)$ = $|x_1 - x_2| + |y_1 - y_2|$




$$|x_1 - x_2| + |y_1 - y_2|$$

- **Hamming distance** is used for categorical variables. In simple terms it tells us if the two categorical variables are same or not.

Hemming distance between categorical variables Large and Medium = 1

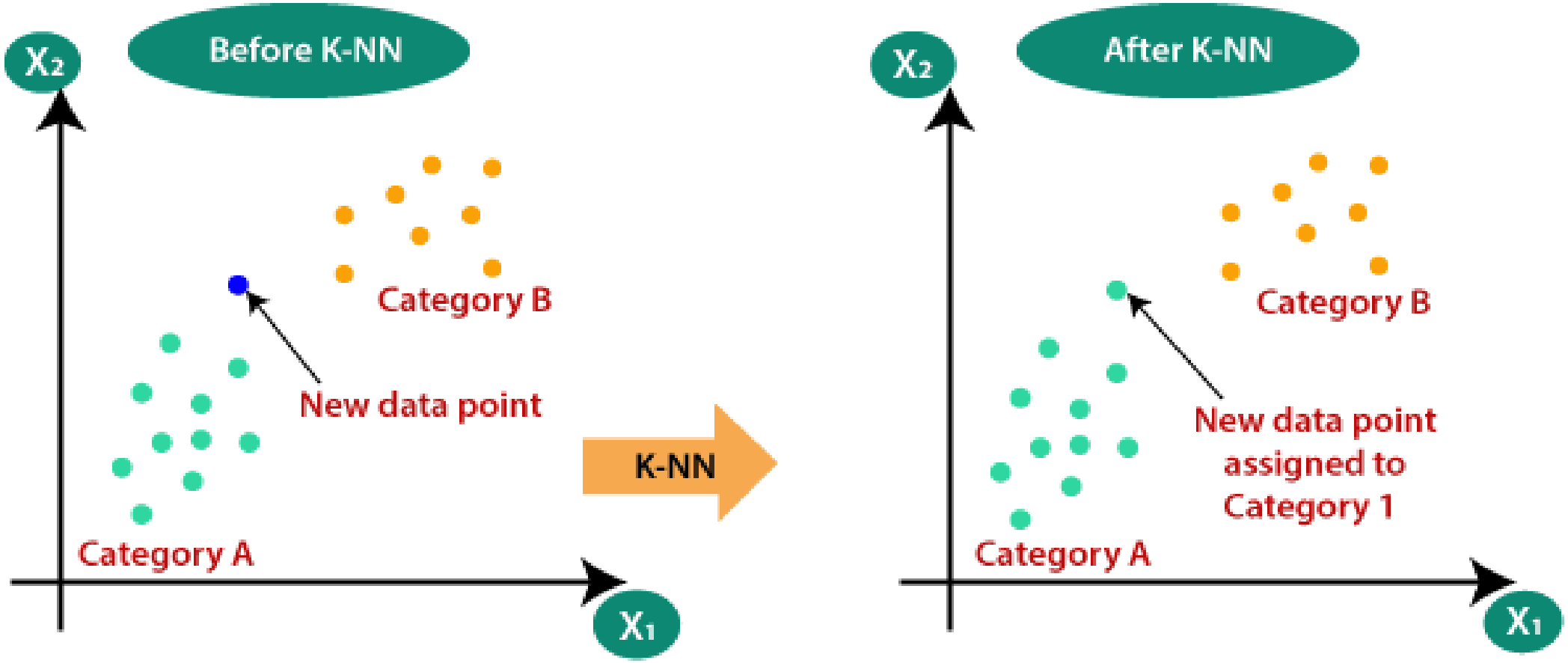
Hemming distance between categorical variables Large and Large = 0

- **Minkowski distance** is the used to find distance similarity between two points. When p=1, it becomes Manhattan distance and when p=2, it becomes Euclidean distance.

Minkowski Distance between $(x_1, y_1)$ and $(x_2, y_2)$ = $(|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$

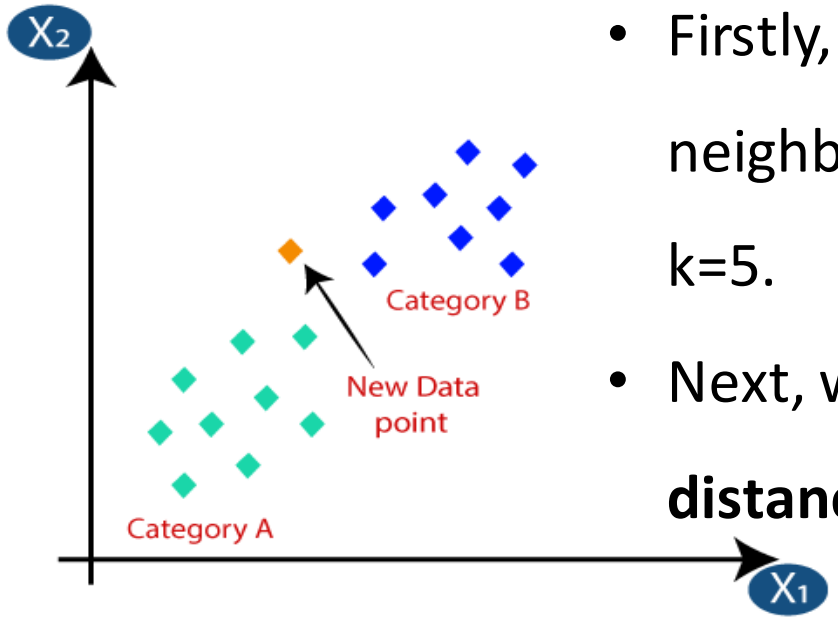
*p is any real value. Usually set to a value between 1 and 2*

- **Cosine similarity for NLP**

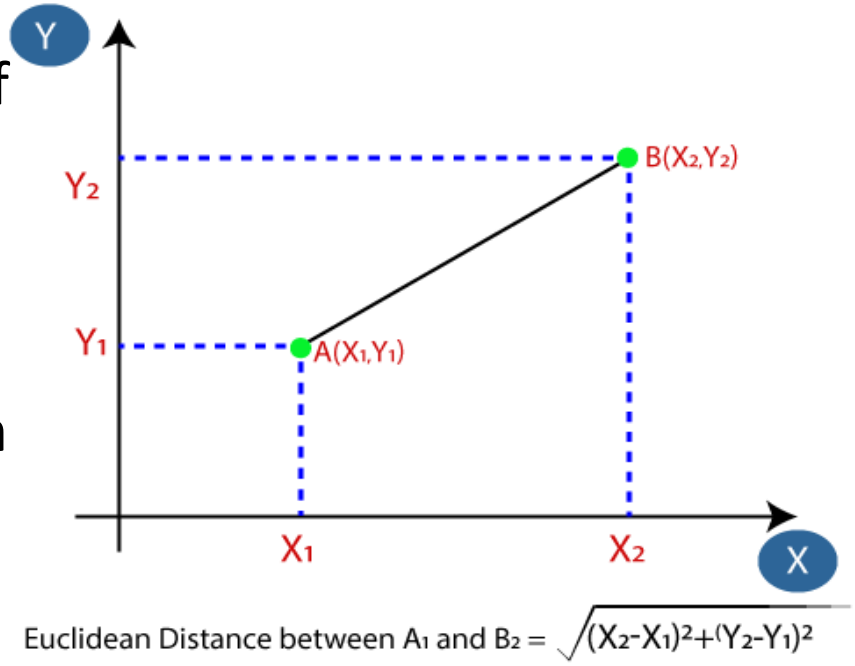

Before K-NN
X2
Category B
New data point
Category A
X1
K-NN
After K-NN
X2
Category B
New data point
assigned to
Category 1
Category A
X1

# *How does KNN work?*

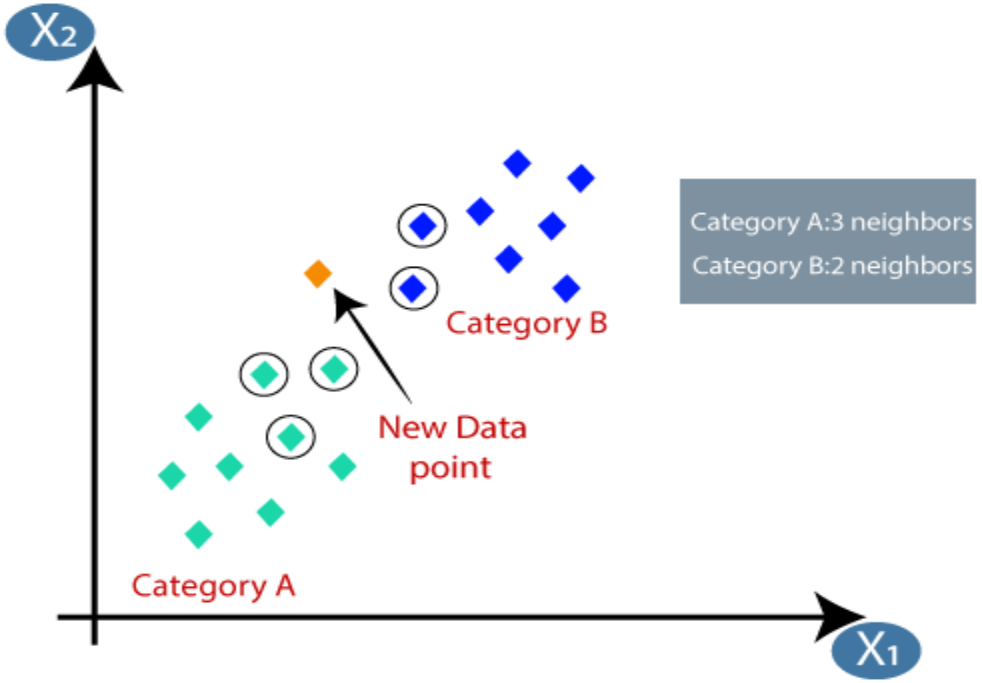
- We have age and experience in an organization along with the salaries.
- We want to predict the salary of a new candidate whose age and experience is available.
- **Step 1: Choose a value for K**. K should be an odd number.
- **Step2: Find the distance of the new point to each of the training data**.
- **Step 3: Find the K nearest neighbors to the new data point**.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.
- For regression, value for the **new data point will be the average of the k neighbors**.




- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points.




- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.
- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.



# *What are the Pros and Cons of KNN?*

## Pros

- Simple algorithm and hence easy to interpret the prediction
- Non parametric, so makes no assumption about the underlying data pattern
- used for both classification and Regression
- Training step is much faster for nearest neighbor compared to other machine learning algorithms.

## Cons

- KNN is computationally expensive as it searches the nearest neighbors for the new point at the prediction stage
- High memory requirement as KNN has to store all the data points
- Prediction stage is very costly
- Sensitive to outliers, accuracy is impacted by noise or irrelevant data.

# Applications

- Credit rankings
- Customer churn
- Loan bankruptcy
- Optical character recognition (OCR)
- Image recognition

- **KNN for Regression :**

The prediction is based on the **mean or the median** of the K-most similar instances.

- **KNN for Classification :**

The output can be calculated as the class with the **highest frequency** from the K-most similar instances.

The class with the **most votes** is taken as the prediction.

# Curse of Dimensionality

- KNN works well with a small number of input variables (p), but struggles when the number of inputs is very large.
- Each input variable can be considered a dimension of a p-dimensional input space.
- For example, if you had two input variables x1 and x2, the input space would be 2-dimensional.
- In high dimensions, points that may be similar may have very large distances.
- All points will be far away from each other and our intuition for distances in simple 2 and 3-dimensional spaces breaks down.

# Best Prepare Data for KNN

**Rescale Data**: KNN performs much better if all of the data has the same scale.

Normalizing your data to the range [0, 1] is a good idea.

**Lower Dimensionality**: KNN is suited for lower dimensional data.

- You can try it on high dimensional data (hundreds or thousands of input variables) but be aware that it may not perform as well as other techniques.
- KNN can benefit from feature selection that reduces the dimensionality of the input feature space

# KNN for regression

- Mean/median of the 'y' variable in 'k' most similar instances

**w.r.t KNN, what happens if I have categorical data:**

**Label Encoder:** Size: S, M, L **< LE >**Size: 1,2,3

**One Hot Encoder:**

Gender: M, F **< OHE >**Create 2 new columns Gender_M and Gender_F

**Dummy coder**

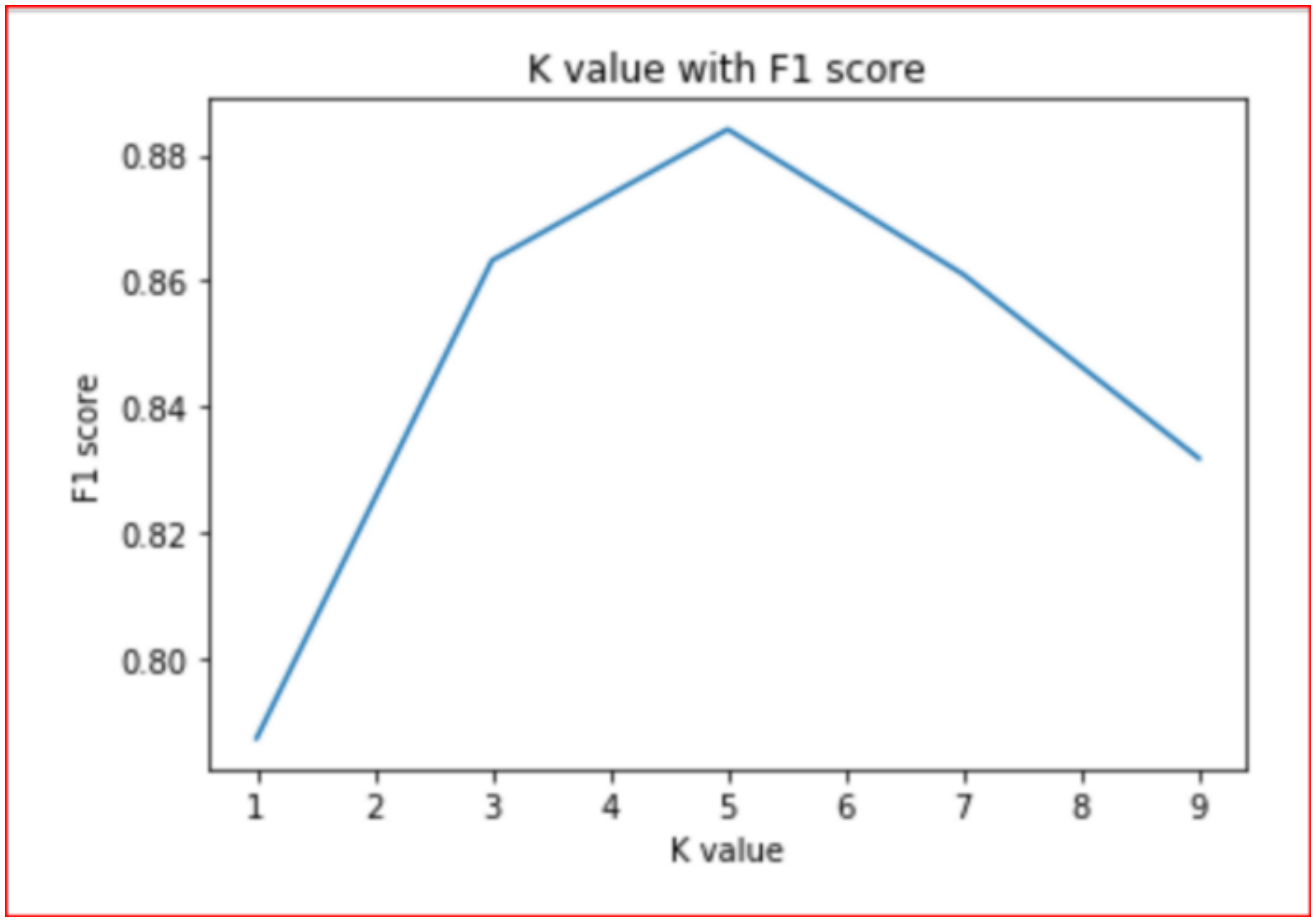
- Is similar to One Hot Encoder except that it tries to minimize the columns by 1
- Car Type: H, S, M **< DC >**
- Create only 2 new columns Car_Type_H and Car_Type_S

## *Two questions arise if we try to understand K-NN :*

1. ***How do we decide the value of K?***
2. ***Which value is the nearest value i.e which distance metrics can be used?***

## *How do we chose the value of K?*

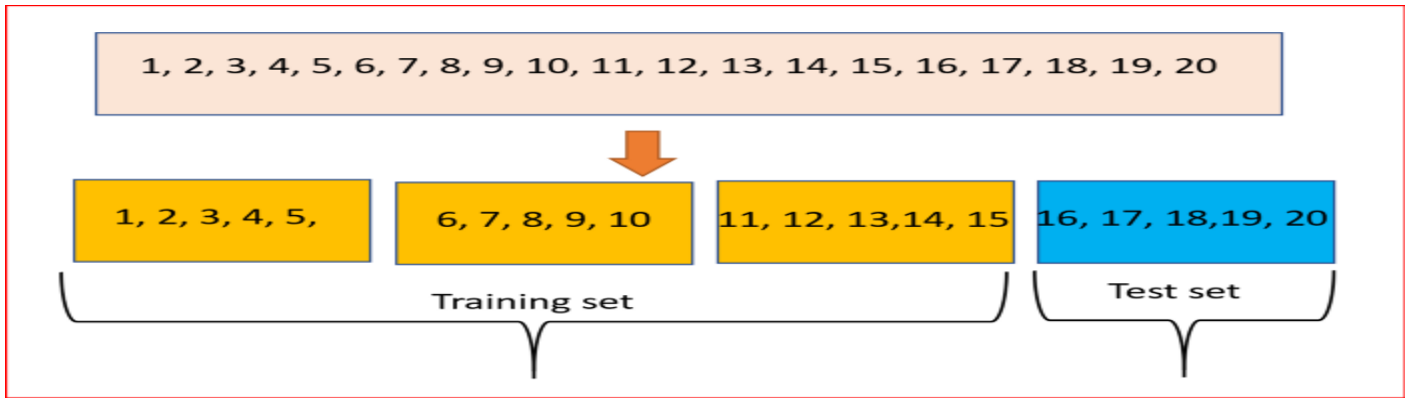
- Choice of K has a drastic impact on the results we obtain from KNN.
- we can take the test set and plot the accuracy rate or F1 score against different values of K.
- We see a high error rate for test set when K=1. we can conclude that model over fits.
- For a high value of K, we see that the F1 score starts to drop. The test set reaches a minimum error rate when k=5. This is very similar to the elbow method used in K-means.
- Value of K at the elbow of test error rate gives us the optimal value of K.


K value with F1 score
F1 score
0.88
0.86
0.84
0.82
0.80
1
2
3
4
5
6
7
8
9
K value

- We can evaluate accuracy of KNN classifier using **K fold cross validation.**

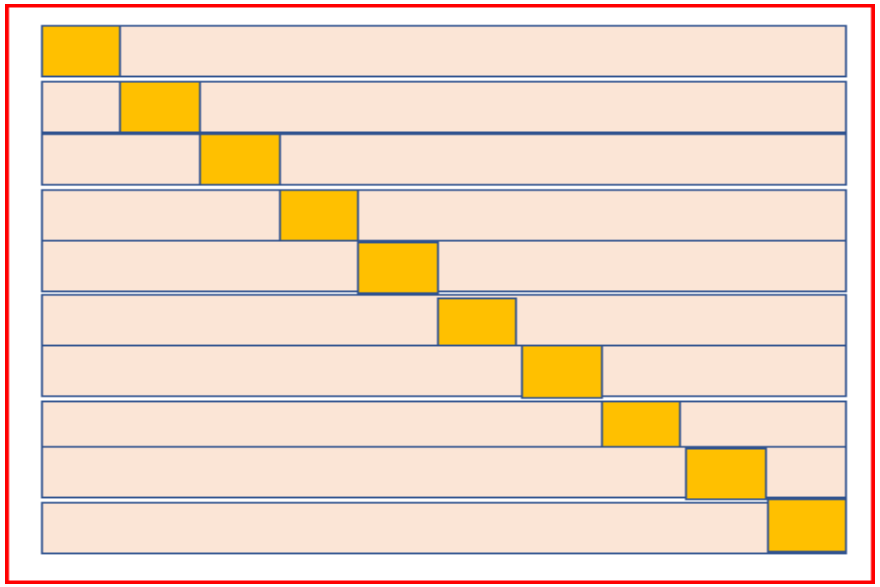
***What is cross-validation?***

- Cross-validation is a statistical technique which involves partitioning the data into subsets, training the data on a subset and use the other subset to evaluate the model's performance.
- To reduce variability we perform multiple rounds of cross-validation with different subsets from the same data.
- We combine the validation results from these multiple rounds to come up with an estimate of the model's predictive performance.
- Cross-validation will give us a more accurate estimate of a model's performance



# K fold cross validation

- This technique involves ***randomly dividing the dataset into k groups or folds*** of approximately equal size.
- The ***first fold is kept for testing*** and the **model is trained on k-1 folds**.
- The process is repeated K times and each time different fold or a different group of data points are used for validation.



# EXAMPLE

| Age | Loan | Default | Distance | |
|---|---|---|---|---|
| 25 | $40,000 | N | 102000 | |
| 35 | $60,000 | N | 82000 | |
| 45 | $80,000 | N | 62000 | |
| 20 | $20,000 | N | 122000 | |
| 35 | $120,000 | N | 22000 | 2 |
| 52 | $18,000 | N | 124000 | |
| 23 | $95,000 | Y | 47000 | |
| 40 | $62,000 | Y | 80000 | |
| 60 | $100,000 | Y | 42000 | 3 |
| 48 | $220,000 | Y | 78000 | |
| 33 | $150,000 | Y | 8000 | 1 |
| | | | | |
| **48** | **$142,000** | **?** | | |

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

# KNN for Classification

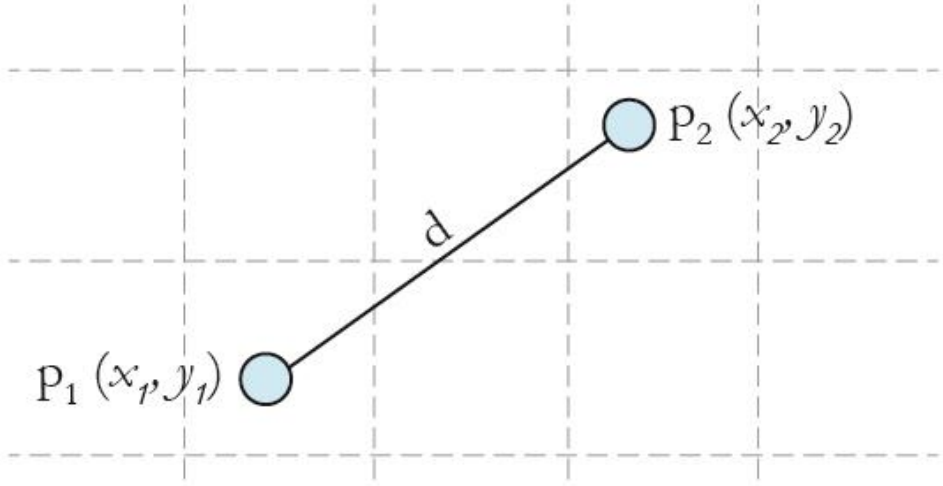
- Let's consider 10 'drinking items' which are rated on two parameters on a scale of 1 to 10.
- The two parameters are **"sweetness" and "fizziness"**.
- **Sweetness"** determines the perception of the sugar content in the items.
- **"Fizziness"** ascertains the presence of bubbles in the drink due to the carbon dioxide content in the drink. The ratings of few items look somewhat as:

| Ingredient | Sweetness | Fizziness | Type of Drink |
|---|---|---|---|
| Monster | 8 | 8 | Energy booster |
| ACTIV | 9 | 1 | Health drink |
| Pepsi | 4 | 8 | Cold drink |
| Vodka | 2 | 1 | Hard drink |

This will be determined by calculating distance.

## Calculating Distance

- Now, calculating distance between 'Maaza' and its nearest neighbors ('ACTIV', 'Vodka', 'Pepsi' and 'Monster') requires the usage of a distance formula, the most popular being Euclidean distance formula



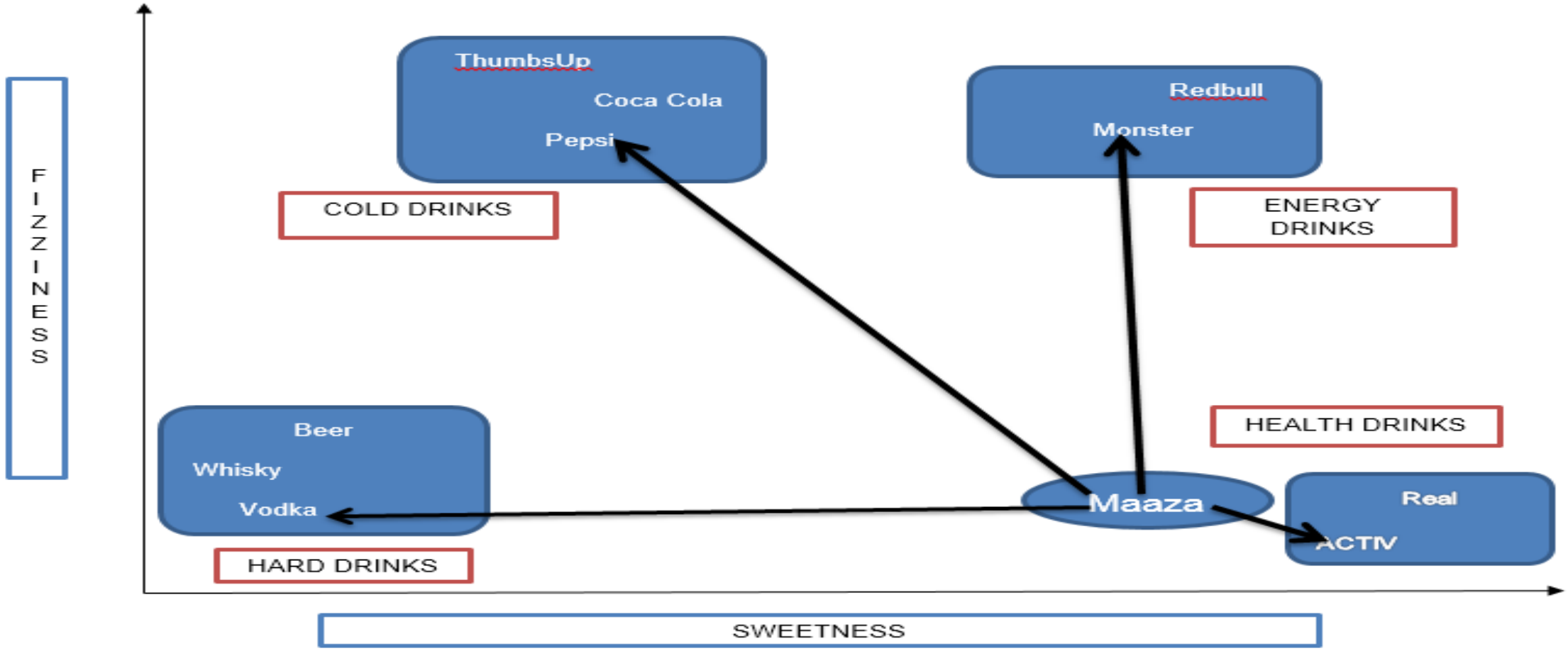

Euclidean distance (d) $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

**The question here is, to which group would 'Maaza' fall into?**

Using the co-ordinates of Maaza (8,2) and Vodka (2,1), the distance between 'Maaza' and 'Vodka' can be calculated as:

*dist(Maaza,Vodka) = Sqrt ((8-2)^2+(2-1)^2)= Sqrt(36+1)=6.08*

| Ingredient | Sweetness | Fizziness | Type of Drink | Distance to Maaza |
|---|---|---|---|---|
| Monster | 8 | 8 | Energy booster | 6.08 |
| ACTIV | 9 | 1 | Health drink | 1.41 |
| Pepsi | 4 | 8 | Cold drink | 7.21 |
| Vodka | 2 | 1 | Hard drink | 6.08 |




- **From the above figure, it is clear we have bucketed the 10 items into 4 groups namely, 'COLD DRINKS', 'ENERGY DRINKS', 'HEALTH DRINKS' and 'HARD DRINKS'.**
- **If k=1, the algorithm considers the nearest neighbor to Maaza i.e, ACTIV;**
- **if k=3, the algorithm considers '3' nearest neighbors to Maaza to compare the distances (ACTIV, Vodka, Monster) – ACTIV stands the nearest to Maaza.**

# How to measure the performance of classification algorithm ?

# Accuracy, Confusion Matrix, Precision/Recall

- In machine learning, evaluating the performance of classifiers is more tricky to understand than evaluating regression-based models.
- Imagine you're a **security guard for a store**.
- You work half the time and another security guard(your coworker) works the other half.
- Duty is **a security guard is trying to do is prevent theft.**
- For simplicity, let's say that 90% of people who visit the store are there to shop and not steal anything.
- The other 10% of visitors, however, are looking for a five finger discount. (Apparently this store makes people want to steal.)

- Let’s first imagine this scenario. Your coworker is terrible at the job and despite the statistic that 10% of people who enter the store try to steal something, he almost never stops anyone!
- The **threshold** he uses to act is very, very high. Let’s say, in this case, threshold values range from 0 to 100 and that your coworker is at 99, meaning that he has to be very, very certain that someone is stealing for him to act.
- Even a very high suspicion will not lead him to act. He must be certain.
- As a result, when he is on watch, almost the entire 10% of people who come to steal get away with it.
- The manager calls him in and tells him that there is a lot of theft under his watch.
- What does your coworker say? “Well, I’m still about 90% accurate!”

- Here lies the problem with using **accuracy** as a performance measure with imbalanced classes in a classification task.
- In this case, the two classes of visitor are ‘thief’ and ‘not thief’, where ‘thief’ only accounts for 10% of the people who visit the store.
- So, yes, your coworker is right. Even when he does nothing, he is still 90% right when it comes to classifying who is stealing and who is not simply because 90% of visitors do not steal.
- A ferret dressed up in a security guard outfit would be about as accurate.

- Your boss gets angry and says he prefers different ways to evaluate security guard performance.
- He says one way he measures performance is through a **Confusion Matrix**.
- In fact, through video footage of shoppers, he has created a confusion matrix of your coworker's performance with a sample of 1,000 people who entered the store during his watch.
- Below portrays a confusion matrix.
- The rows represent the actual values for each class while the columns represent the predicted values for each class.

| n=# of observations | Predicted: NO (not a thief) | Predicted: YES (a thief) |
|---|---|---|
| Actual: NO (not a thief) | True Negative (TN) | False Positive (FP) |
| Actual: YES (a thief) | False Negative (FN) | True Positive (TP) |

- The NO label corresponds to ‘not thief’ visitors and the YES label corresponds to ‘thief’ visitors.
- Let’s go cell by cell in the matrix and clarify some terminology, starting with the top left cell.
- When the security guard predicts that a person belongs to the ‘not thief’ class (Predicted: NO) and the prediction is correct (the visitor was not a thief), it is called a **True Negative** (TN).

| n=# of observations | Predicted: NO (not a thief) | Predicted: YES (a thief) |
|---|---|---|
| Actual: NO (not a thief) | True Negative (TN) | False Positive (FP) |
| Actual: YES (a thief) | False Negative (FN) | True Positive (TP) |

- When the security guard predicts that a visitor belongs to the 'thief' class (Predicted: YES), but the person belongs to the 'not thief' class (incorrect prediction), it is called a **False Positive** (FP).
- When the security guard predicts that a visitor is a 'not thief', but the person is a 'thief' visitor (incorrect prediction), it is called a **False Negative** (FN).
- Finally, when the security guard predicts that a visitor is a 'thief', and the prediction is correct, it is called a **True Positive** (TP).

- he correctly predicts that 894/1000 people belong to the ‘not thief’ class.
- But the matrix he allowed 105 people to steal something from the store since he will not act unless he’s absolutely sure the person is trying to steal. In one case, he actually saw someone steal right in front of him and acted for once. This is the true positive value of 1 in the matrix. (Note that since this is a sample, there will be some randomness in values, which is why 894/1000 is less than 90%.)

| n=1000 | Predicted: NO (not a thief) | Predicted: YES (a thief) |
|---|---|---|
| Actual: NO (not a thief) | 894 | 0 |
| Actual: YES (a thief) | 105 | 1 |

# How to evaluate my Classification Model results

- when we assess classification model results, we should not only focus on prediction results but also need to consider the prediction probability and prediction ability on each class.
- Top 5 important metrics to evaluate your classification model.

1. Confusion Matrix
2. *Accuracy, Recall, Precision*
3. *F1 Score*
4. *Receiving Operating Characteristics(ROC),Area under the Curve(AUC)*
5. *Log Loss*

# Confusion Matrix

- Confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the true values are known.

- Ex: there are 165 patients , in reality 105 patients have the disease and 60 patients don’t have disease. The classifier predicted 'yes' 110 patents and 'no' 55 patients

| Prediction | Actual Value: Positive(1) | Actual Value: Negative(0) |
|---|---|---|
| Positive(1) | True Positive(TP) | False Positive(FP) |
| Negative(0) | False Negative(FN) | True Negative(TN) |

| n=165 | Predicted: NO | Predicted: YES |
|---|---|---|
| Actual: NO | 50 | 10 |
| Actual: YES | 5 | 100 |

- **TP (True Positive):** These are the cases in which we predicted Yes and they do have the disease
- **TN (True Negative):** We predicted No and they don’t have the disease
- **FP (False Positive):** We predicted Yes, but they don’t actually have disease (Type I)
- **FN (False Negative):** We predicted No, but they actually do have the disease (Type II)

**Accuracy:** Overall, how often is the classifier correct?

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

(TN+TP)/N = (50+100)/165=0.9091

**Misclassification rate (Error rate):**Overa

(FP+FN)/N = (10+5)/165=0.0909

**Error = 1 -Accuracy**

| n=165 | **Predicted: NO** | **Predicted: YES** | |
|---|---|---|---|
| **Actual: NO** | TN = 50 | FP = 10 | 60 |
| **Actual: YES** | FN = 5 | TP = 100 | 105 |
| | 55 | 110 | |

**True Positive Rate(Sensitivity/Recall):**

- When it is actually yes, how often does it predict yes
- TP/Actual yes = 100/105=0.9524

**False Positive Rate:** When its actually No, how often does it predict yes.

FP/Actual No = 10/60=0.1667

**Specificity:** When its actually no, how often does it predict No

TN/Actually No = 50/60=0.8333

Specificity = 1 -False Positive Rate

| n=165 | Predicted: NO | Predicted: YES | |
|---|---|---|---|
| Actual: NO | TN = 50 | FP = 10 | 60 |
| Actual: YES | FN = 5 | TP = 100 | 105 |
| | 55 | 110 | |

**Prevalence:** How often does the yes condition actually occur in our data?

Actual Yes/Total = 105/165=0.6364

- **When to use Accuracy?**
- Accuracy is a good measure of how the overall model performs.
- However, it is not telling you the performance in each category and thus you may miss important information if you purely look at accuracy.
- It is best to used **when number of cases in different categories are similar or used together with other metrics**.
- **Recall**
- Recall measures how much percentage of real **positive cases** are correctly identified.
- Recall = TP/(TP+FN)

- **When to use Recall?**

- Recall is used when the **aim is to capture maximum number of positive cases**. For example, during the recent Covid-19 situation, the government wants to track all the infected cases in the community. If you have built an model to predict whether the person is infected with Covid-19 based on symptoms, Recall will be an important metric to be measured.

- **Precision**

- Precision measures that among the cases predicted to be positive, how much percentage of them are really positive

- Precision = TP/(TP+FP)

- **When to use Precision?**
- Precision is used when you want to be **very accurate in measuring positive cases**.
- It always **conflicts with Recall because the more positive cases you want to capture, the lower the standard you will put to be classified as positive cases**.
- In the Covid-19 cases example, if you want to capture more infected cases, you may want to include all cases that have slight symptoms that could just be because of normal flu.

# What is F1 Score?

- Considering the conflicting feature between precision and recall, F1 Score is created to **have a balanced metric between recall and precision**. It is the Harmonic mean of recall and precision.
- **why do we need to have F1 Score?**

$$\text{F1 Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- You may want to ask: why we do not simply average Precision and Recall? Well, if the distribution of positive and negative cases are very uneven, the average may not be a good representation of model performance.

| | |
|---|---|
| TP=1 | FP=99 |
| FN=0 | TN=900 |

- In the example above, there is only 1 real positive case and the model captures it.

  However, among the 100 cases identified to be positive, only 1 of them is really positive.

- Thus, recall=1 and precision=0.01.

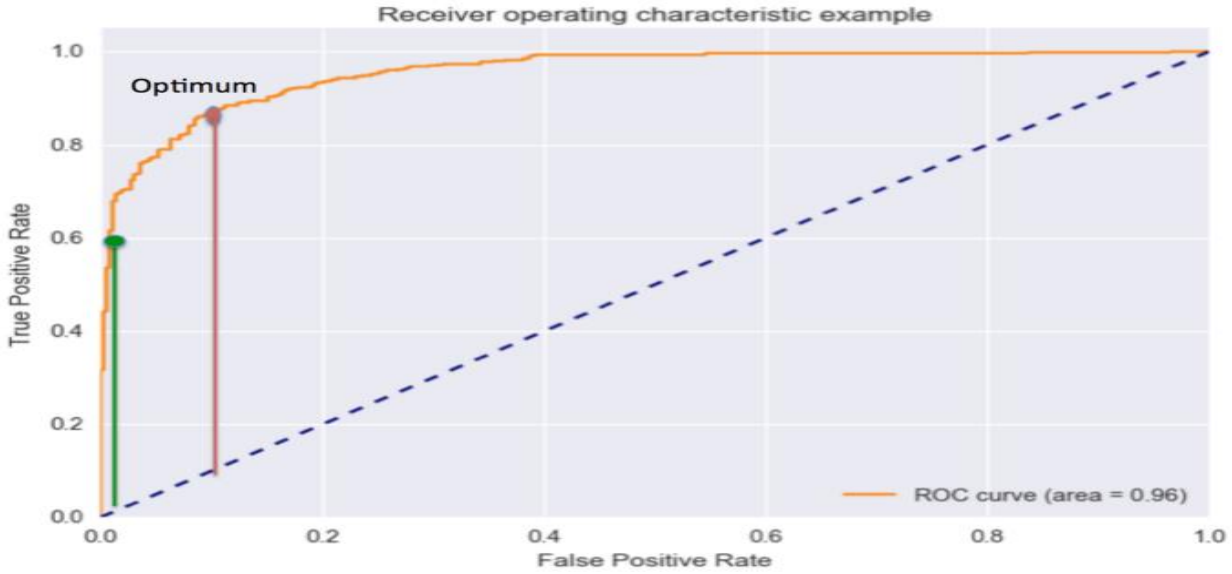
- The average between the two is 0.505 which is clearly not a good representation of how bad the model is. F1 score= 2*(1*0.01)/(1+0.01)=0.0198 and this gives a better picture of how the model performs.

# Receiving Operating Characteristics(ROC), Area under the Curve(AUC)

- Receiving Operating Characteristics(ROC), Area under the Curve(AUC)
- AUC is the area under the ROC curve and it is a good measure on two things:
- How well the model can separate the two classes(positive and negative)
- How accurate is the model identify for different categories(like whether it identify group A to positive correctly)?
- **Here we need to understand two metrics first:**
- True Positive Rate(TPR): TPR actually equals to Recall which is among the truly positive cases, how much percentage of them are captured in the model correctly
- False Positive Rate(NPR): This measures among the truly negative cases, how much percentage of them are actually false positive

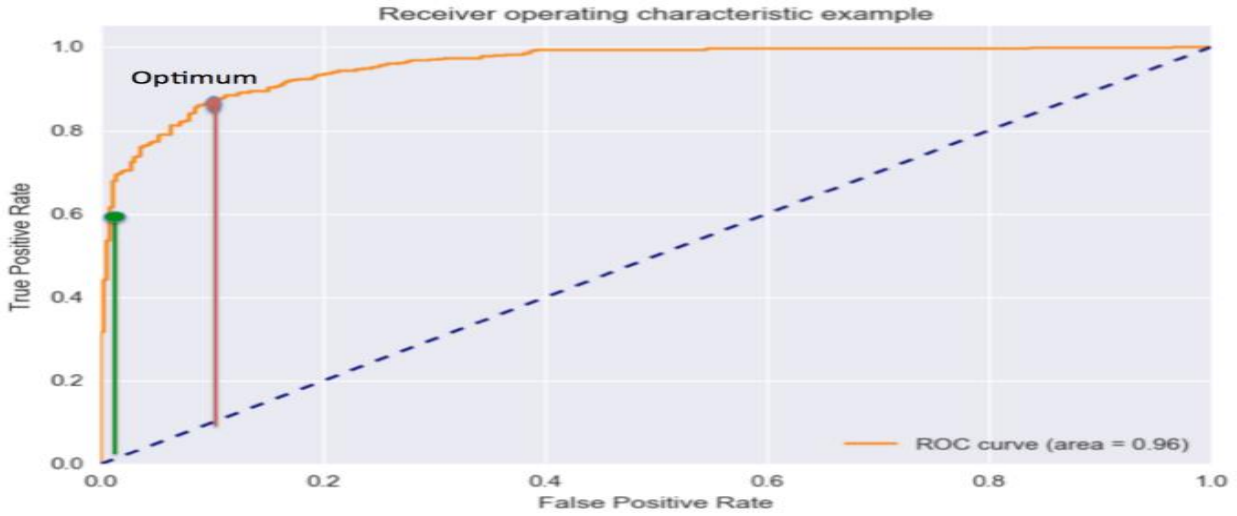
# What is ROC curve then?

- TPR and FPR are positively correlated. This is because when you want to capture more positive cases in your c misclassify some of them and NPR ir
- **ROC curve is TPR plotted against NP the better the separation between c**




- For classification problem on 2 class(either 1 or 0), the probability of you blindly guessing is 50%.
- The blue dotted line shows the curve TPR and FPR when you blindly guess the category and for that diagonal line, area under the curve(AUC) is 0.5.

- Each point on the curve represents a different cut off point between TPR and FPR. Example, if I cannot tolerate any false positive rate (<0.01), then the true positive rate I can reach is around 0.6(see the green dot on the graph). If I loosen my criteria a bit and I only need to control FPR below 0.1, then my TPR can reach 0.9(see the red dot on the graph).



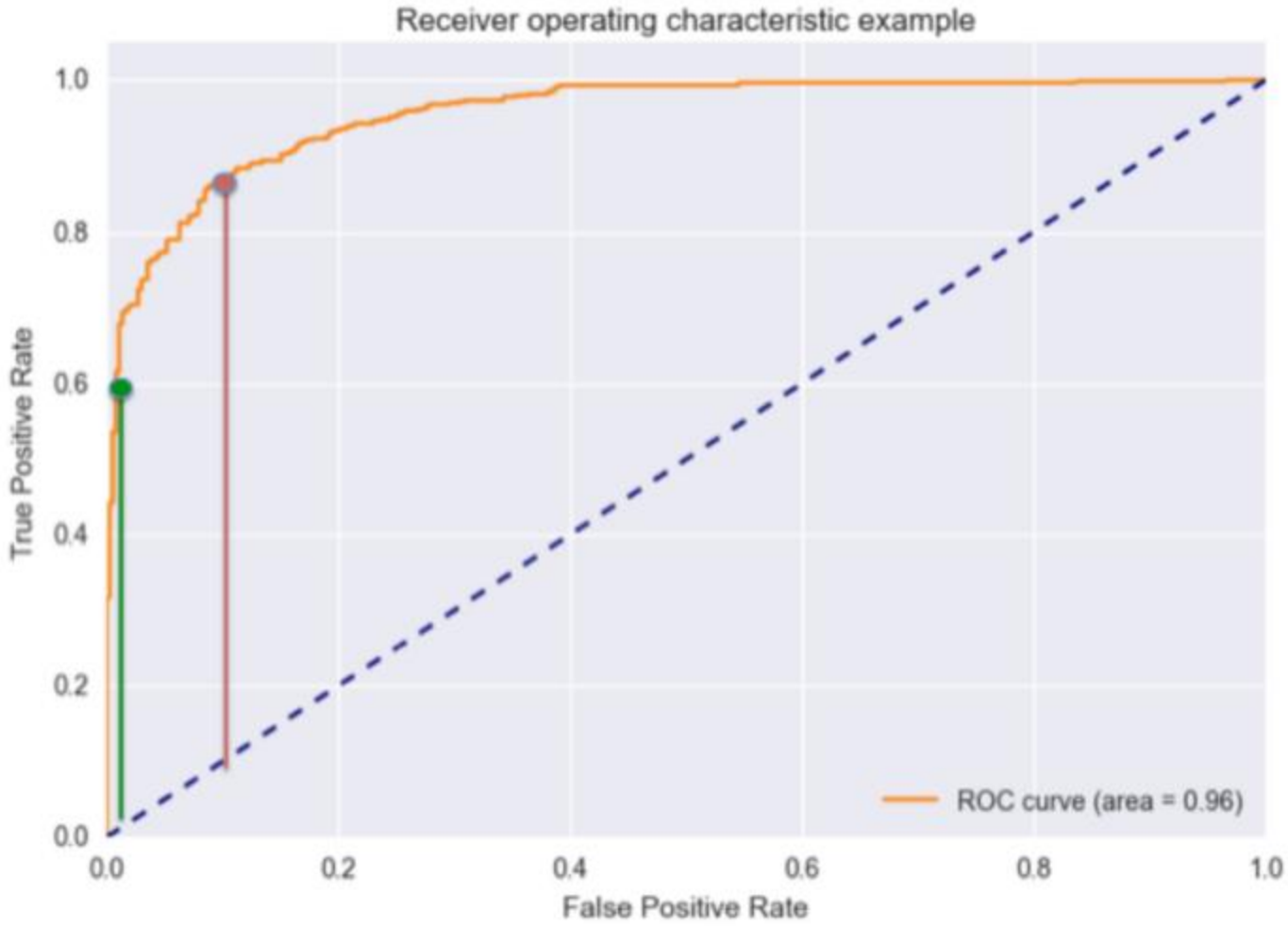

- The selection of threshold is based on different business requirement and usually, the best choice is to maximise the difference between TPR and FPR and on the graph is represented by the maximum vertical distance between orange and blue dotted line(see the red line). Our prediction example has a pretty good AUC(0.96) and optimised point is around (0.1,0.9)

- When you build a classification model like decision tree and you want to classify maybe whether the stock will increase or decrease tomorrow based on the input.
- The classification model will first calculate the probability of increase or decrease based on historical value you provided.
- After that, based on the **threshold**, it will decide whether the result is increase or decrease.
- Yes, the keyword here is threshold. **Different threshold creates different TPR and FPR.**
- **They represent different points and form up the ROC curve.**
- You can choose the output to be 'Increase' if more than 50% chance of increase based on historical data but you can also increase the threshold and only show 'Increase' if more than 90% chance.
- If you choose 90% confidence instead of 50%, you will be more confident that the stocks you choose to show 'Increase' is very likely to increase but you may miss out some potential stocks.

- Why I plot the blue dotted line on the graph?
- As we know the bigger the area under the curve(AUC), the better the classification. The ideal or perfect curve is vertical line from (0,0) to (0,1) and then to (1,1) which means the model can 100% separate positive and negative cases.
- However, if you pick randomly for each case, TPR and FPR should increase at the same rate. The blue dotted line shows the curve TPR and FPR when you blindly guess positive or negative for each case and for that diagonal line, area under the curve(AUC) is 0.5.

- What will happen to the TPR, FPR and ROC curve if I change my threshold?
- Look at the two dots on the ROC curve. Green dot has a very high threshold it means only if you are 99% sure then you can classify the case as positive. Red dot has a relatively lower threshold and it means you can classify the case to be positive if you are 90% confident.
- Whats the movement of TPR and FPR from green dot to red dot?
- Both TPR and FPR increase. When you reduce the threshold, more positive cases can be identified, thus TP increases and TP/(TP+FN) also increases. On the other side, inevitably you will wrongly classify some negative cases to be positive due to the lowering of threshold and so FP & FP/(FP+TN) also increases.
- We can see that TPR and FPR are positively correlated and you have to balance between maximising capture of positive cases and minimising wrong classification of negative cases.

# How to choose optimal point on the ROC Curve



- it is hard to define the optimal point because you should choose the most suitable threshold for the business case.
- However, **the general rule is to maximise (TPR-FPR) which in the graph is represented by the vertical distance between orange and blue dotted line**.
- We can see from the chart above that among the three points, red point has the biggest TPR-FPR and should be considered as the optimal point.
- Moving from red point to green point, the threshold gets more strict. However, big number of positive cases are not captured from the model with less than proportionate decrease in wrongly classified cases.
- Moving from red point to blue point, the threshold is getting more lenient and adding more wrongly classified positive points than correctly identified positive points.

# Why Area under the ROC curve is a good metric for evaluation of classification model?

- A good metric of machine learning model should display true and consistent prediction ability of the model. It means if I change the testing dataset it should not have a very different score.
- **ROC curve does not only consider the classification results but also the prediction probability of all classes.** For example, if the output is classified correctly based on 51% probability, then it is very likely to be wrongly-classified if you use another testing dataset. In addition, ROC curve also considers the performance across different threshold and so it is a comprehensive metric to assess **how good the separation of the cases** in different groups.
- **What is acceptable AUC score for a classification model?**
- As I have shown before, for binary classification problem, if you pick randomly you can achieve 0.5 AUC. Thus, if you are solving binary classification problem, **a reasonable ROC score should>0.5**. A good classification model usually has AUC score >0.9 but it is totally application dependent.

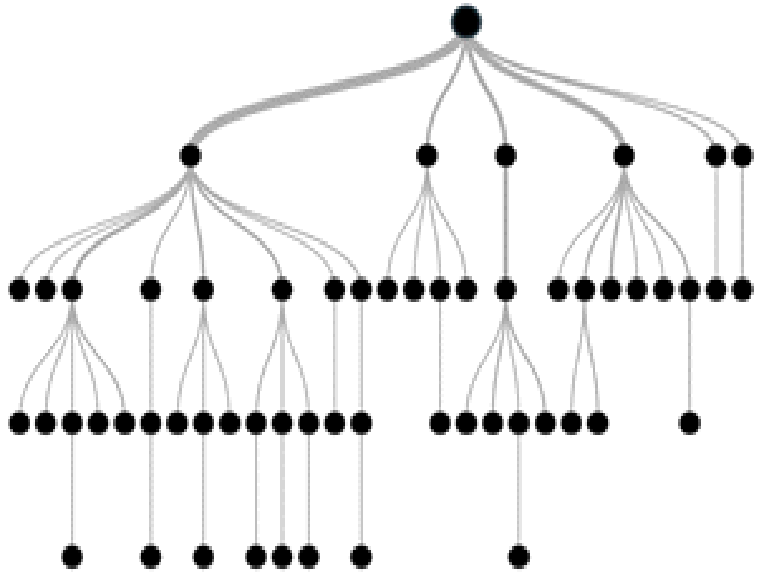
# Decision Tree

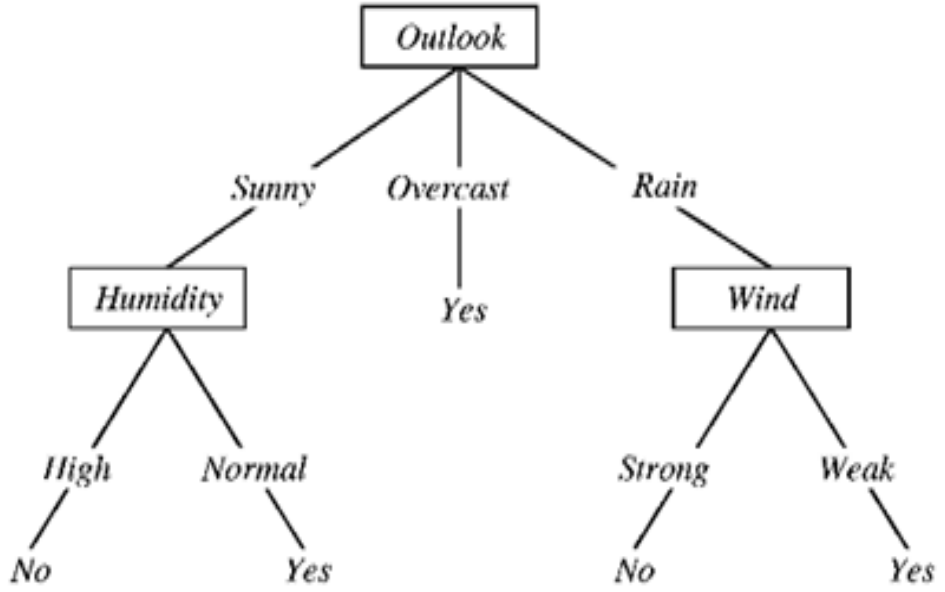
# Introduction

- Decision Tree is mostly used in classification problems and works for both **categorical and continuous input and output variables.**

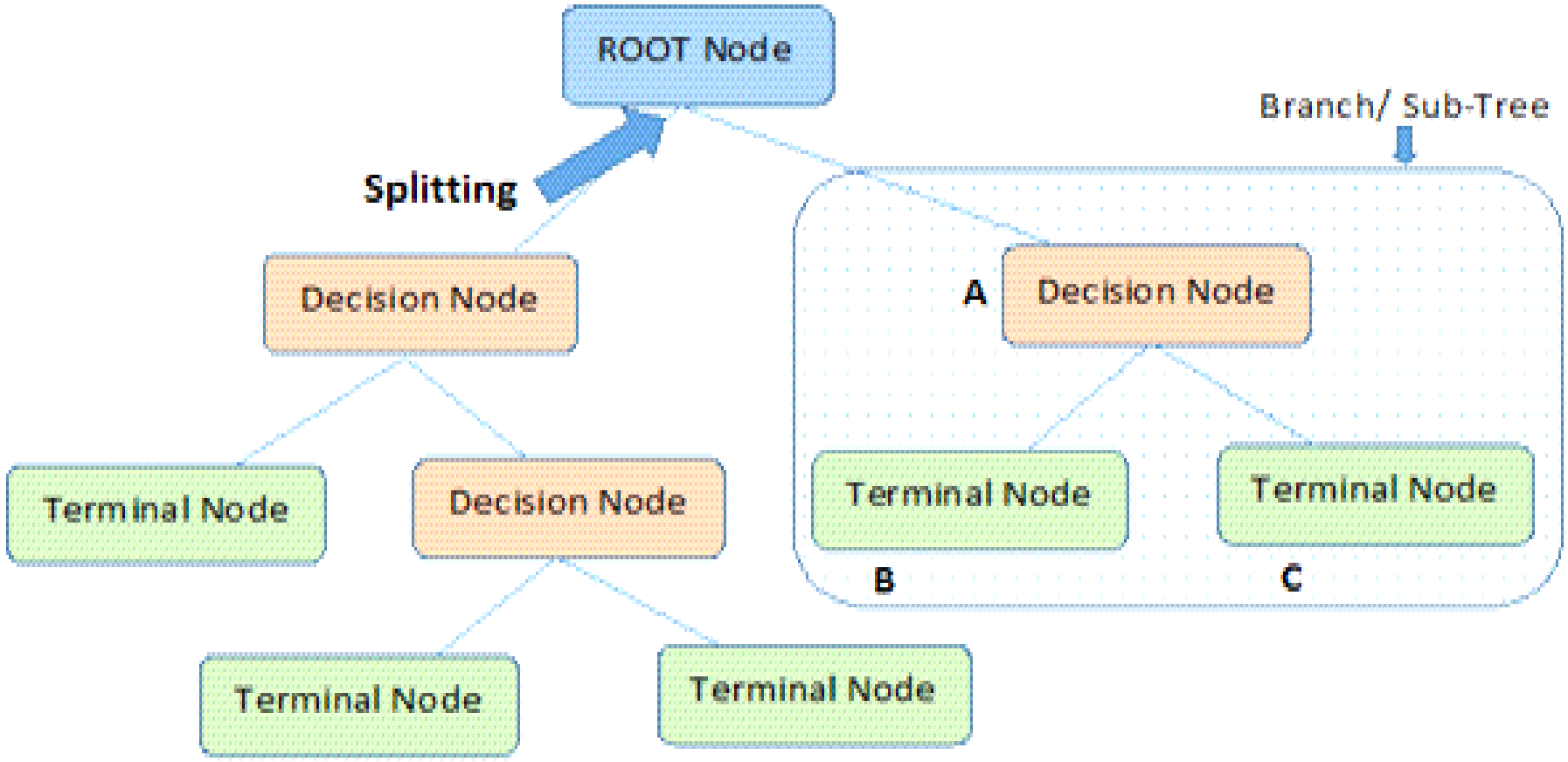
**Divide and conquer (Recursive partitioning):**

it uses the features values to split the data into smaller and smaller subsets of similar classes





# Terminologies related to decision trees



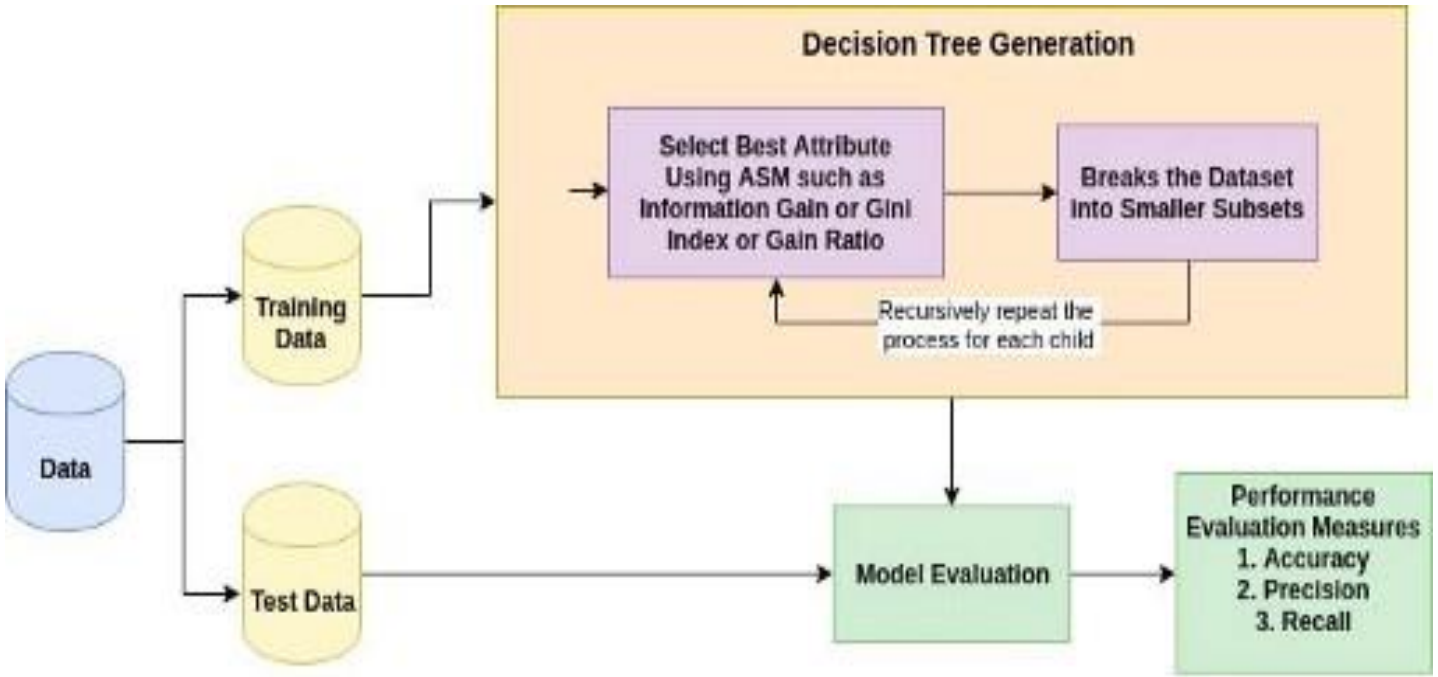

**Note:-** A is parent node of B and C.

# Terminologies related to decision trees:

- **Root Node:** This attribute is used for dividing(population/sample and this further gets divided into two or more homogenous sets) the data into two or more sets. The feature attribute in this node is selected based on Attribute Selection Techniques.
- **Splitting:** Dividing a node into two or more sub-nodes based on if-else conditions.
- **Decision Node:** After splitting the sub-nodes into further sub-nodes, then it is called as the Decision Node
- **Terminal Node:** This is the end of the decision tree where it cannot be split into further sub-nodes.
- **Pruning:** Removing sub-nodes from a decision node is called pruning
- **Branch/Sub-Tree:** A sub section/part of entire tree is called as Branch or Sub Tree
- **Parent and Child node:** A node which is divided into sub-nodes is called parent node of sub nodes whereas sub nodes are called as the child of parent node

# Working of Decision Tree

- The root node feature is selected based on the results from the Attribute Selection Measure(ASM).
- The ASM is repeated until there is a leaf node or a terminal node where it cannot be split into sub-nodes



## Pros

- Easy to understand/explain/interpret
- Data type is not a constraint
- Non parametric method
- Useful for data exploration: Helps to identify important/significant variables***

## Cons

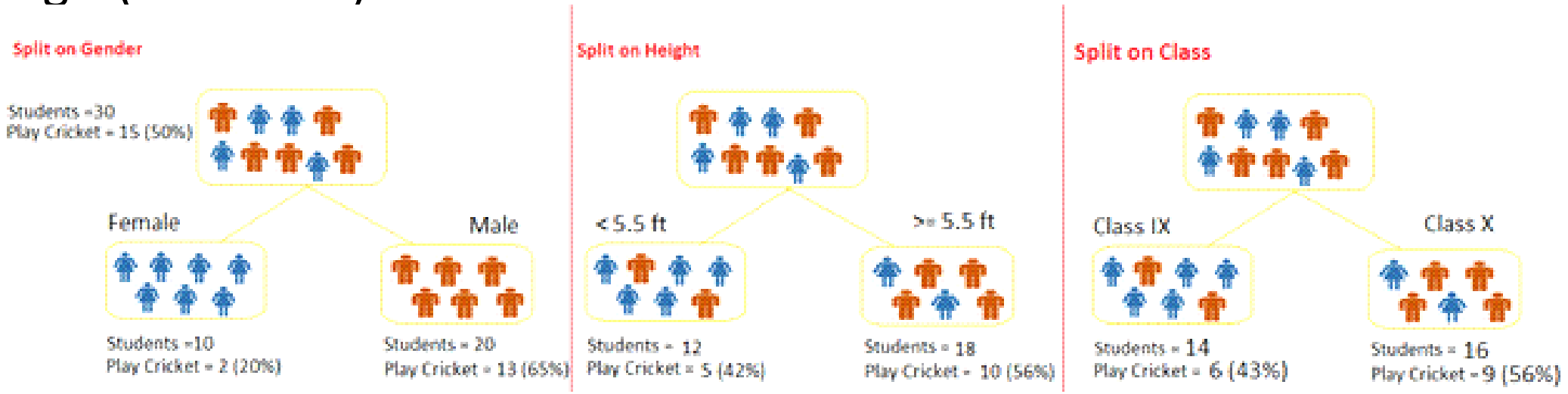
- Over fitting: Greedy Learner
- Relatively the performance goes down for regression problems

# Example

- Sample of 30 students
- 50% willing to play cricket

**3 variables**

- Gender(Boy/Girl)
- Class(IX/X)
- Height(5 ft to 6ft)




Create a model to predict who will play cricket during leisure period

# Attribute Selection Methods

- Attribute Subset Selection Measure is a technique used in the data mining process for data reduction. The data reduction is necessary to make better analysis and prediction of the target variable.
- **The two main ASM techniques are**
- Gini index
- Information Gain(ID3)
- We also use Chisquare test as a ASM

# Gini index

- If we select two items from a population at random then they must be of same class and probability of this 1 if population is pure
- It works with categorical target variables: "Success" or "Failure"
- It performs only on binary splits
- Higher the value of GINI higher the homogenity
- CART uses GINI index method to create **binary splits**

**Steps to calculate**

- Calculate gini for sub nodes: p^2 + q^2
- Calculate gini for split using weighted gini score of each node of split

**Split on gender:**

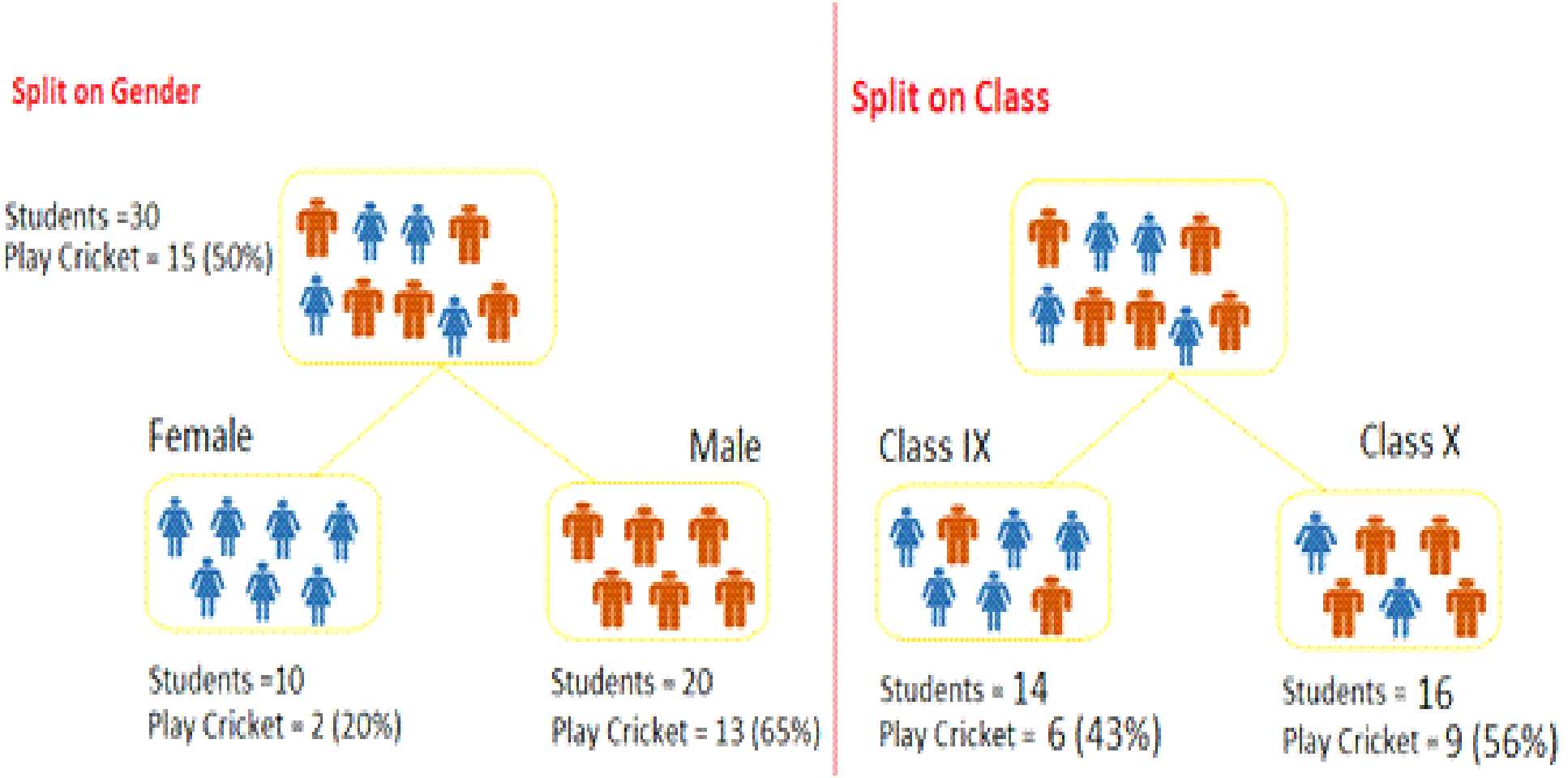
- Gini for sub node: Female: (0.2)^2 + (0.8)^2=0.68
- Gini for sub node: Male: (0.65)^2 + (0.35)^2=0.545
- Calculate weighted gini for split on gender: (10/30)*0.68 + (20/30)*0.55=**0.5933**

**Split on Class:**

- Gini for sub node: Class IX: (0.43)^2 + (0.57)^2=0.5098
- Gini for sub node: Class X: (0.56)^2 + (0.44)^2=0.5072
- Calculate weighted gini for split on class: (14/30)*0.51 + (16/30)*0.51=**0.51**

**We can see that GINI score for split on GENDER is higher than split on CLASS.**

**Hence node split will take place using GENDER**

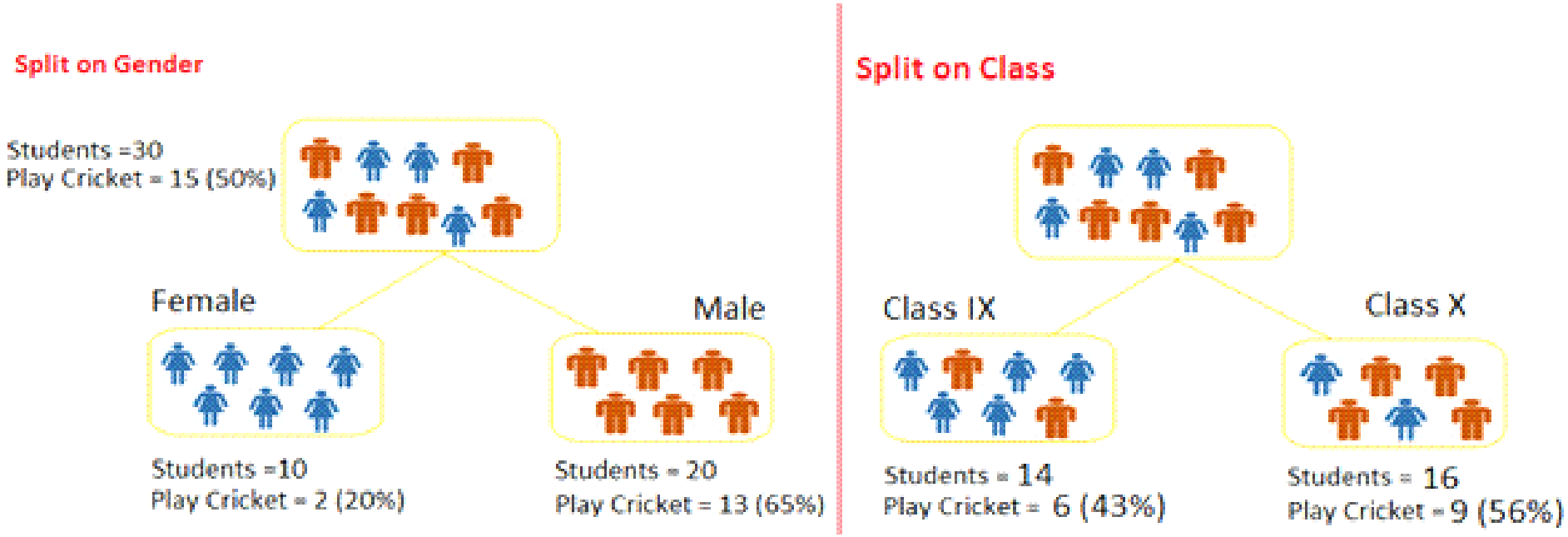

Split on Gender
Students =30
Play Cricket = 15 (50%)
Female
Male
Students =10
Play Cricket = 2 (20%)
Students = 20
Play Cricket = 13 (65%)
Split on Class
Class IX
Class X
Students = 14
Play Cricket = 6 (43%)
Students = 16
Play Cricket = 9 (56%)

# Chi Square:

- It is an algorithm to find out the statistical significance between the differences of sub nodes and parent node
- It works with categorical target variables
- It can perform two or more splits
- Higher the value of chi square higher the statistical significance
- Chi square of each is calculated by **((actual -expected)^2 /expected)^1/2**
- It generates a tree called CHAID (Chi square automatic interaction detector)

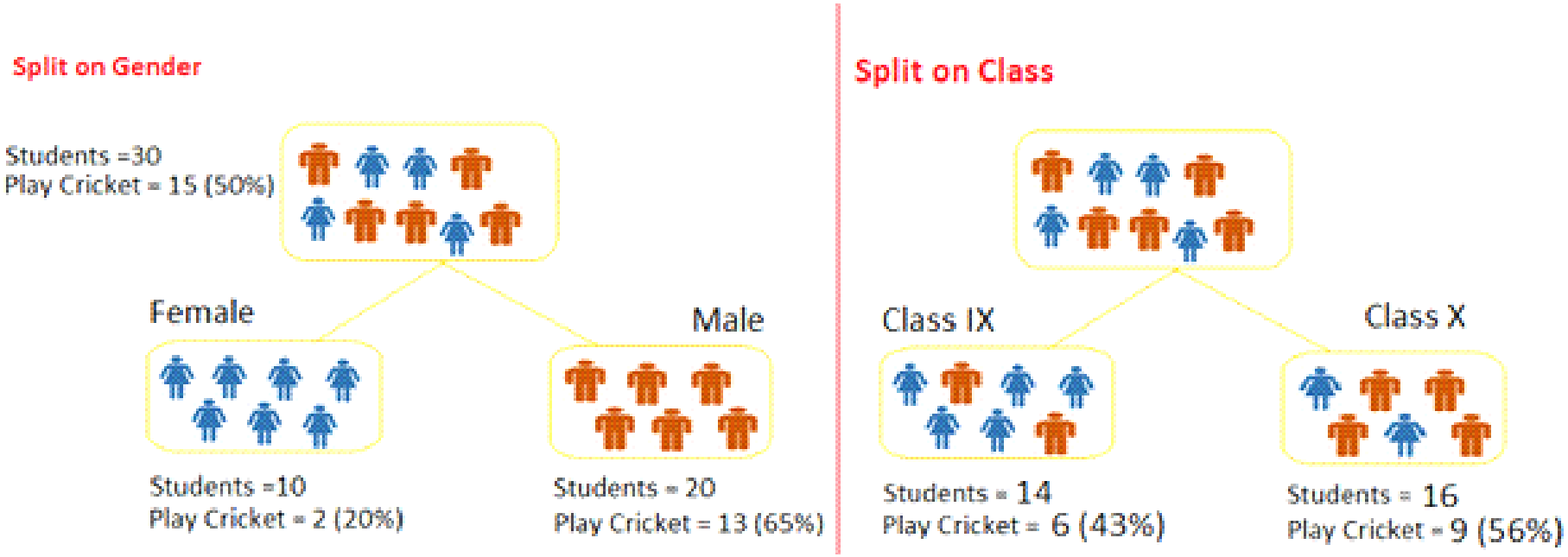
**Steps to calculate:**

- Calculate chi-square for individual nodes
- Calculate chi-square of split using SUM OF ALL CHI_SQUARE OF SUCCESS AND FAILURE OF EACH NODE OF SPLIT




## Split on gender

| node | Play cricket | Not played cricket | total | Expected play cricket | Expected not to play cricket | Deviation on play cricket | Deviation on not to play cricket | Chi square | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Play cricket | Not to Play cricket |
| Female | 2 | 8 | 10 | 5 | 5 | -3 | 3 | 1.34 | 1.34 |
| Male | 13 | 7 | 20 | 10 | 10 | 3 | -3 | 0.95 | 0.95 |




## Split on class

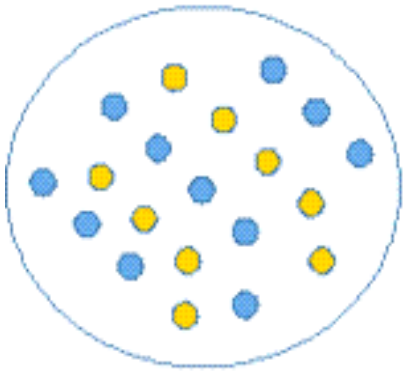
| node | Play cricket | Not played cricket | total | Expected play cricket | Expected not to play cricket | Deviation on play cricket | Deviation on not to play cricket | Chi square | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Play cricket | Not to Play cricket |
| IX | | | | | | | | | |
| X | | | | | | | | | |

# Information gain

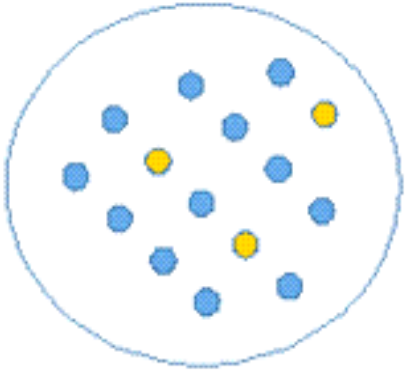
- Entropy can be calculated as **-plogp -qlogq**
- Information Gain = 1 -Entropy
- Lower the entropy, best it is (Higher the Information Gain, the best it is)
- A is more impure and B is less impure and C is pure

**Steps to calculate:**

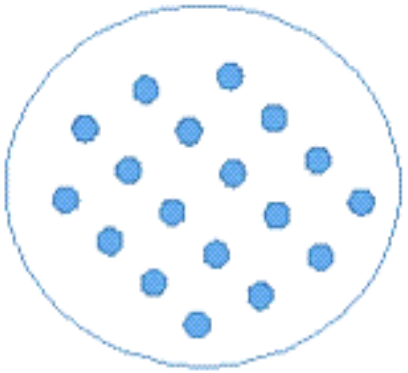
- Calculate the entropy of parent node
- Calculate the entropy of each ind. Node split and
- Calculate weighted average of all sub nodes



A



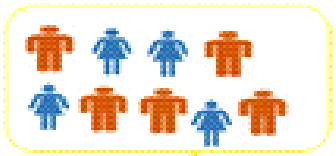
B



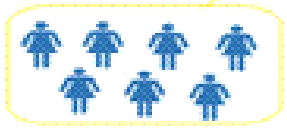
C

**Split on Gender**

Students =30
Play Cricket = 15 (50%)

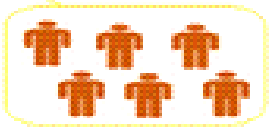


Female



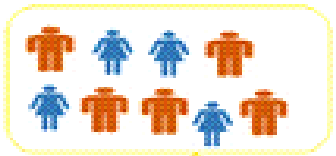
Students =10
Play Cricket = 2 (20%)

Male



Students = 20
Play Cricket = 13 (65%)

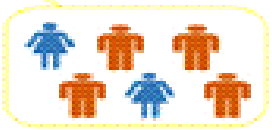
**Split on Class**



Class IX

Students = 14
Play Cricket = 6 (43%)

Class X



Students = 16
Play Cricket = 9 (56%)

# Split on Gender:

- Entropy for parent node = -(15/30)log(15/30) -(15/30)log(15/30) =1

  here '1' shows that it is a impure node

- Entropy for female node = -(2/10)log(2/10) -(8/10)log(8/10) = 0.72
- Entropy for male node = -(13/20)log(13/20) -(7/20)log(7/20) =0.93
- Calculated weighted average = (10/30)*0.72 + (20/30)*0.93=0.86
- Informatio gain 1 - .86=.14

## Split on Class:

- 0.99
- Entropy for split on gender is lowest, so node split will take place on GENDER

# Key parameters of tree modeling:

## Setting constraints on tree size:

- Minimum samples for a node split (DN)
- Minimum samples for a terminal node (TN)
- Maximum depth of a tree (Vertical depth)
- Maximum number of terminal nodes
- Maximum features to consider for split

## Tree pruning:

- Pre pruning
- Post pruning*

# Random forest

The prediction error for any machine learning algorithm can be broken down into three parts:

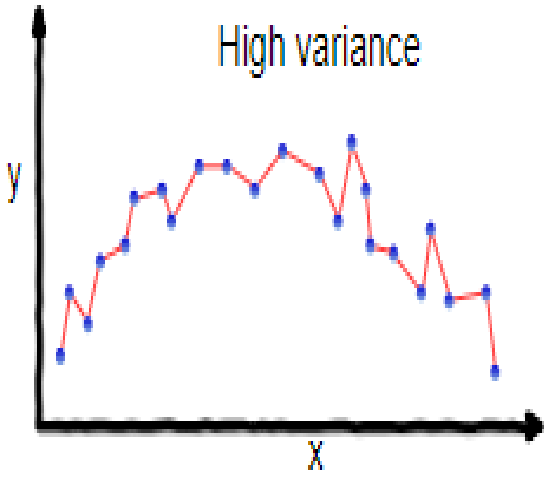
- **Bias Error**
- **Variance Error**
- **Irreducible Error** cannot be reduced regardless of what algorithm is used.
- It is the error introduced from the chosen framing of the problem and may be caused by factors like unknown variables that influence the mapping of the input variables to the output variable.

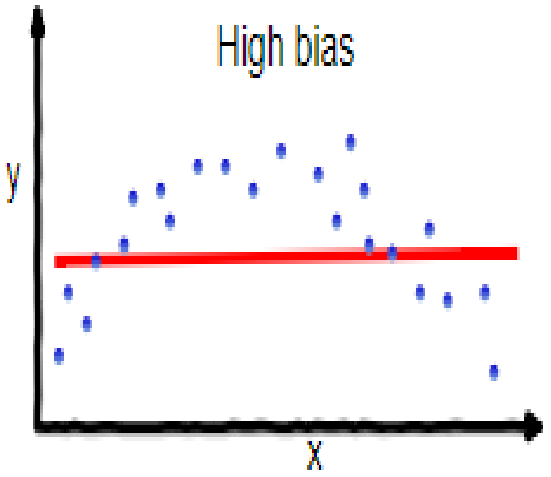
# Bias Error

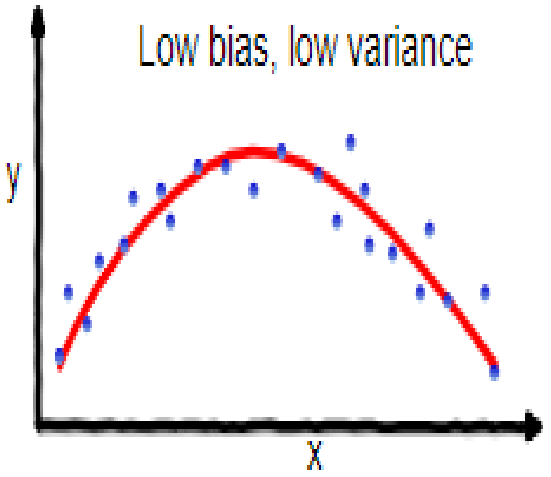
- Bias are the simplifying assumptions made by a model to make the target function easier to learn.
- Generally, linear algorithms have a high bias making them fast to learn and easier to understand but generally less flexible.
- Examples of **low-bias** machine learning algorithms include: Decision Trees, k-Nearest Neighbors and Support Vector Machines.
- Examples of **high-bias** machine learning algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression.

# Variance Error

- Variance is the amount that the estimate of the target function will change if different training data was used.
- The target function is estimated from the training data by a machine learning algorithm, so we should expect the algorithm to have some variance.
- Machine learning algorithms that have a high variance are strongly influenced by the specifics of the training data.
- Examples of **low-variance** machine learning algorithms include: **Linear Regression, Linear Discriminant Analysis and Logistic Regression.**
- Examples of **high-variance** machine learning algorithms include: **Decision Trees, k-Nearest Neighbors and Support Vector Machines.**




**overfitting**

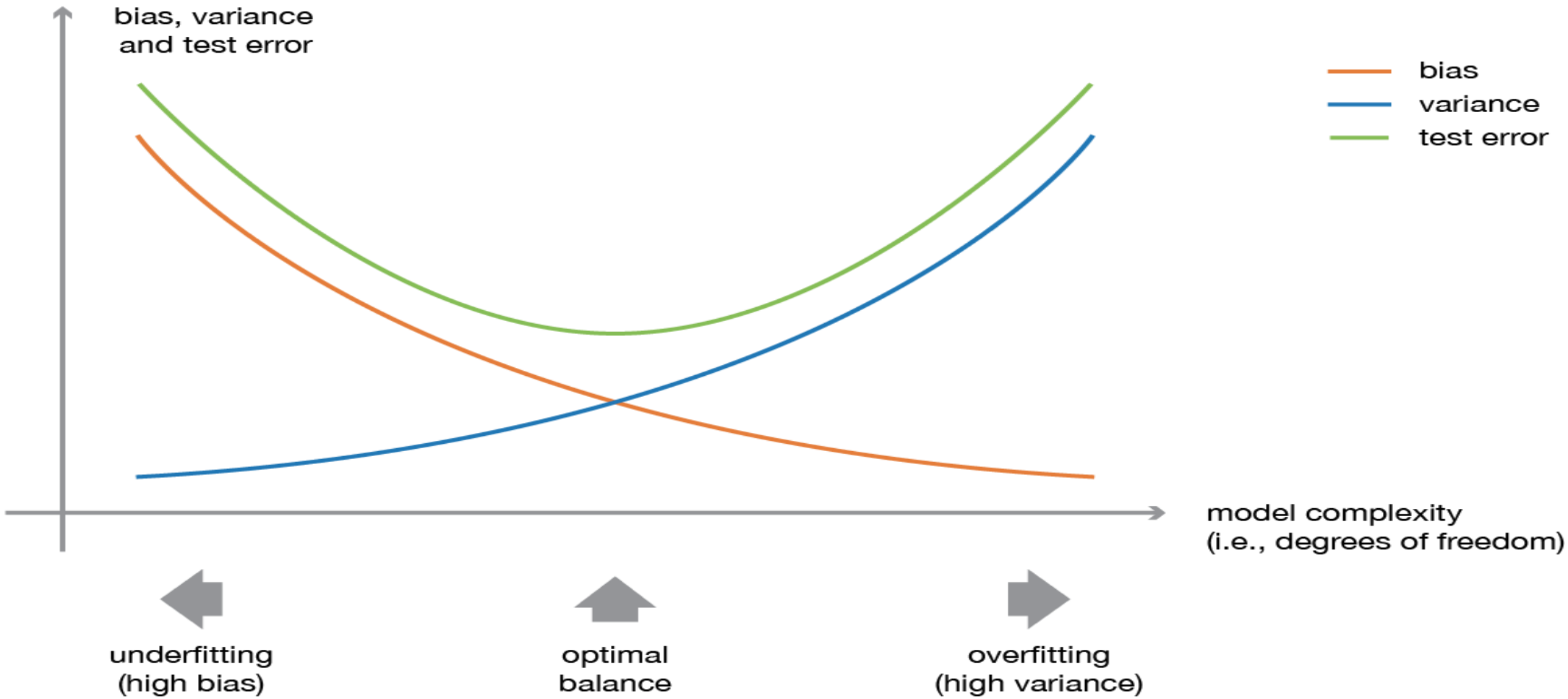
**underfitting**

**Good balance**

- To have any chance to obtain good results depend on many variables of the problem: quantity of data, dimensionality of the space, distribution hypothesis…

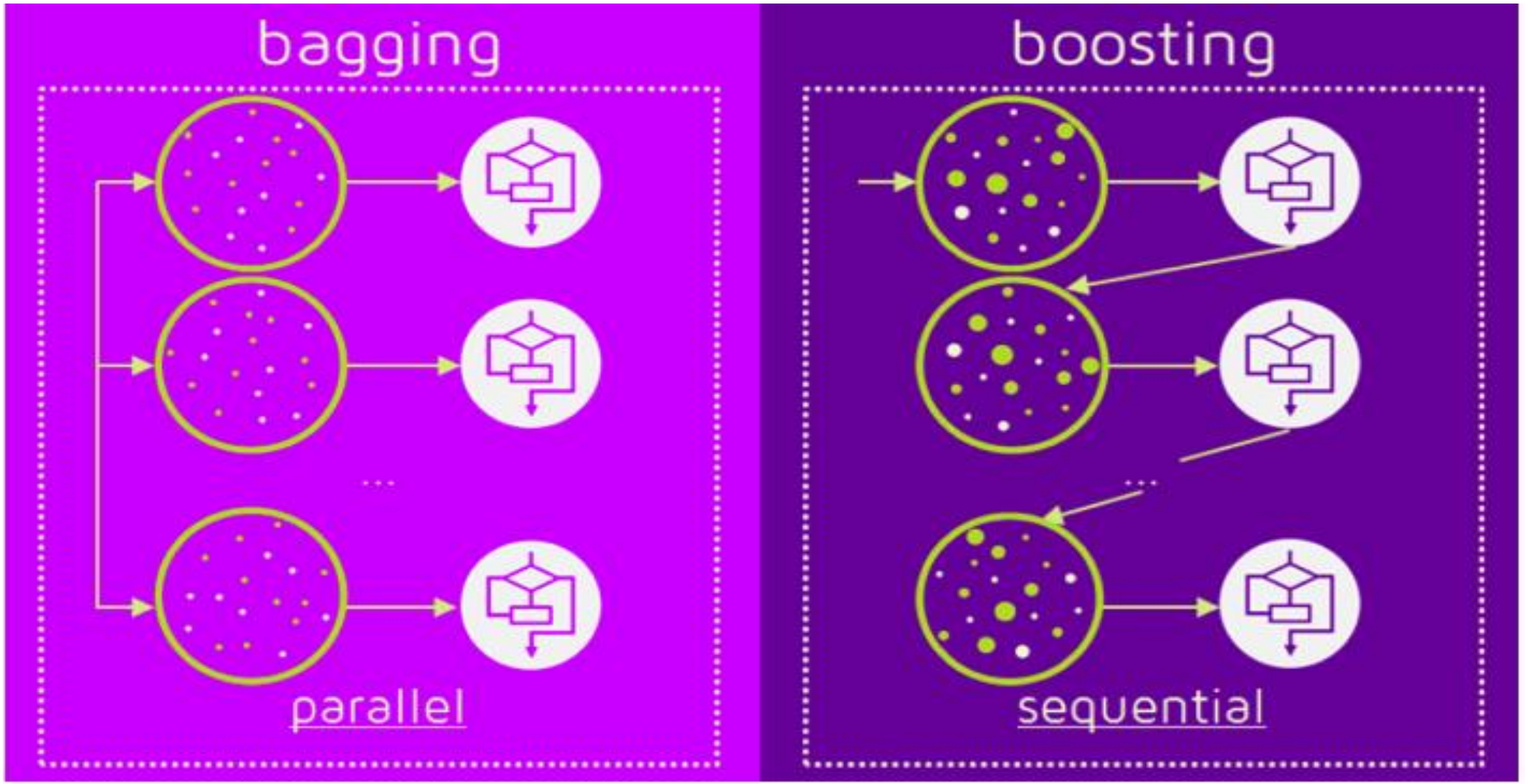
**bias-variance tradeoff**.

- A low bias and a low variance, although they most often vary in opposite directions, are the two most fundamental features expected for a model.
- To “solve” a problem, we want our model to have enough degrees of freedom to resolve the underlying complexity of the data we are working with, but we also want it to have not too much degrees of freedom to avoid high variance and be more robust.


bias, variance
and test error
bias
variance
test error
model complexity
(i.e., degrees of freedom)
underfitting
(high bias)
optimal
balance
overfitting
(high variance)

- **Bagging**: that often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process
- **Boosting**: that often considers homogeneous weak learners, learns them sequentially in a very adaptive way (a base model depends on the previous ones) and combines them following a deterministic strategy
- **Stacking**: that often considers heterogeneous weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions

# Ensemble Techniques



- **Ensemble methods** combine several decision trees classifiers to produce better predictive performance than a single decision tree classifier.
- The main principle behind the ensemble model is that a **group of weak learners come together to form a strong learner**, thus increasing the accuracy of the model.
- **Bagging** is used when the goal is to reduce the variance of a decision tree classifier.
- **Here the objective is to create several subsets of data from training sample chosen randomly with replacement.**
- **Each collection of subset data is used to train their decision trees.**
- As a result, we get an ensemble of different models.
- Average of all the predictions from different trees are used which is more robust than a single decision tree classifier.

# Bagging Steps:

- Suppose if we have **N observations** and **M features** in training data set.
- A sample from training data set is taken **randomly with replacement**.
- A subset of M features are selected randomly and whichever feature gives the best split is used to split the node iteratively. The tree is grown to the largest.
- Above steps are repeated n times and prediction is given based on the aggregation of predictions from n number of trees.

**Advantages:**

- **Reduces over-fitting** of the model.
- Handles **higher dimensionality** data very well.
- Maintains **accuracy for missing data**.

**Disadvantages:** Since final prediction is based on the mean predictions from subset trees, it won't give precise values for the classification and regression model.

# Boosting

- Boosting is used to create a collection of predictors.
- **In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data for errors.**
- **Consecutive trees (random sample) are fit and at every step, the goal is to improve the accuracy from the prior tree.**
- When an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly. This process converts weak learners into better performing model.

**Boosting Steps:** Draw a random subset of training samples d1 without replacement from the training set D to train a weak learner C1

- Draw second random training subset d2 without replacement from the training set and add 50 percent of the samples that were previously falsely classified/misclassified to train a weak learner C2.
- Find the training samples d3 in the training set D on which C1 and C2 disagree to train a third weak learner C3
- Combine all the weak learners via majority voting.

**Advantages:** Supports different loss function (we have used 'binary:logistic' for this example).

- Works well with interactions.

**Disadvantages:** Prone to over-fitting.

- Requires careful tuning of different hyper-parameters.

# Comparison Matrix

| Classification Algorithms | Accuracy | F1-Score |
|---|---|---|
| Decision Tree | 76.67 % | 0.74 |
| Bagging | 77.67 % | 0.77 |
| Boosting | 78.67 % | 0.78 |

**Accuracy: (True Positive + True Negative) / Total Population**

**True Positive:** The number of correct predictions that the occurrence is positive

**True Negative:** The number of correct predictions that the occurrence is negative

**F1-Score: (2 x Precision x Recall) / (Precision + Recall)**

**F1-Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1-Score is usually more useful than accuracy, especially if you have an uneven class distribution.**

**Precision:** When a positive value is predicted, how often is the prediction correct?

**Recall:** When the actual value is positive, how often is the prediction correct?

# Random forest

- Panacea for all the data science problems
- On a funny note, when you cant think of any algorithm use Random Forest
- Versatile ML methods capable of doing

  Classification

  Regression

  Missing values

  Outliers

  Dimensionality Reduction (Variable Importance plot)

# Algorithm of RF:

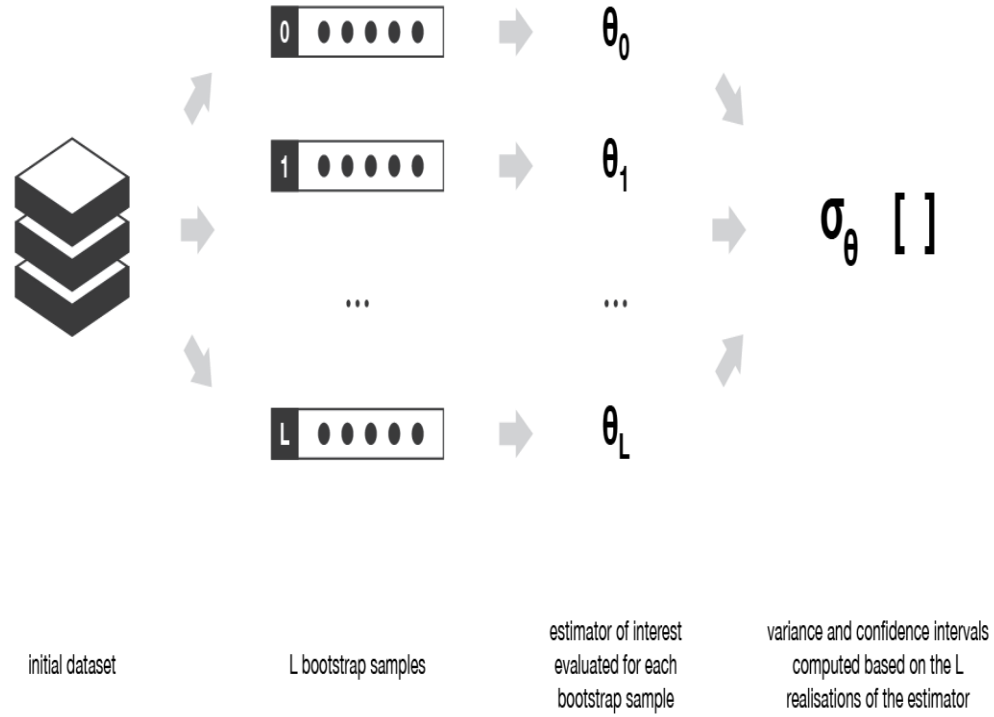
- Random forest us like **bootstrapping** algorithm with decision tree model
- The **random forest** approach is a bagging method where **deep trees**, fitted on bootstrap samples, are combined to produce an output with lower variance.
- 1000 rows and 10 columns
- Consider, as an example, max of 70% data and 60% of variables ==> 700 rows and 6 columns
- In Random Forest, we grow multiple trees as against a single tree in DT model

# How does it work?

- Assume number of cases in training set is 'N', sample of 'n' cases is taken at random but with replacement (n < N)
- If there are 'M' input variables, a number of 'm' number such that m < M is specified at each node
- We will grow trees to full extent on each one of the random subsets
- Aggregate the results(Majority voting for classification and mean for regression)

# Bootstrapping

- This statistical technique consists in generating samples of size B (called bootstrap samples) from an initial dataset of size N by randomly drawing with replacement B observations.

Random forest method is a bagging method with trees as weak learners.

Each tree is fitted on a bootstrap sample considering only a subset of variables randomly chosen.

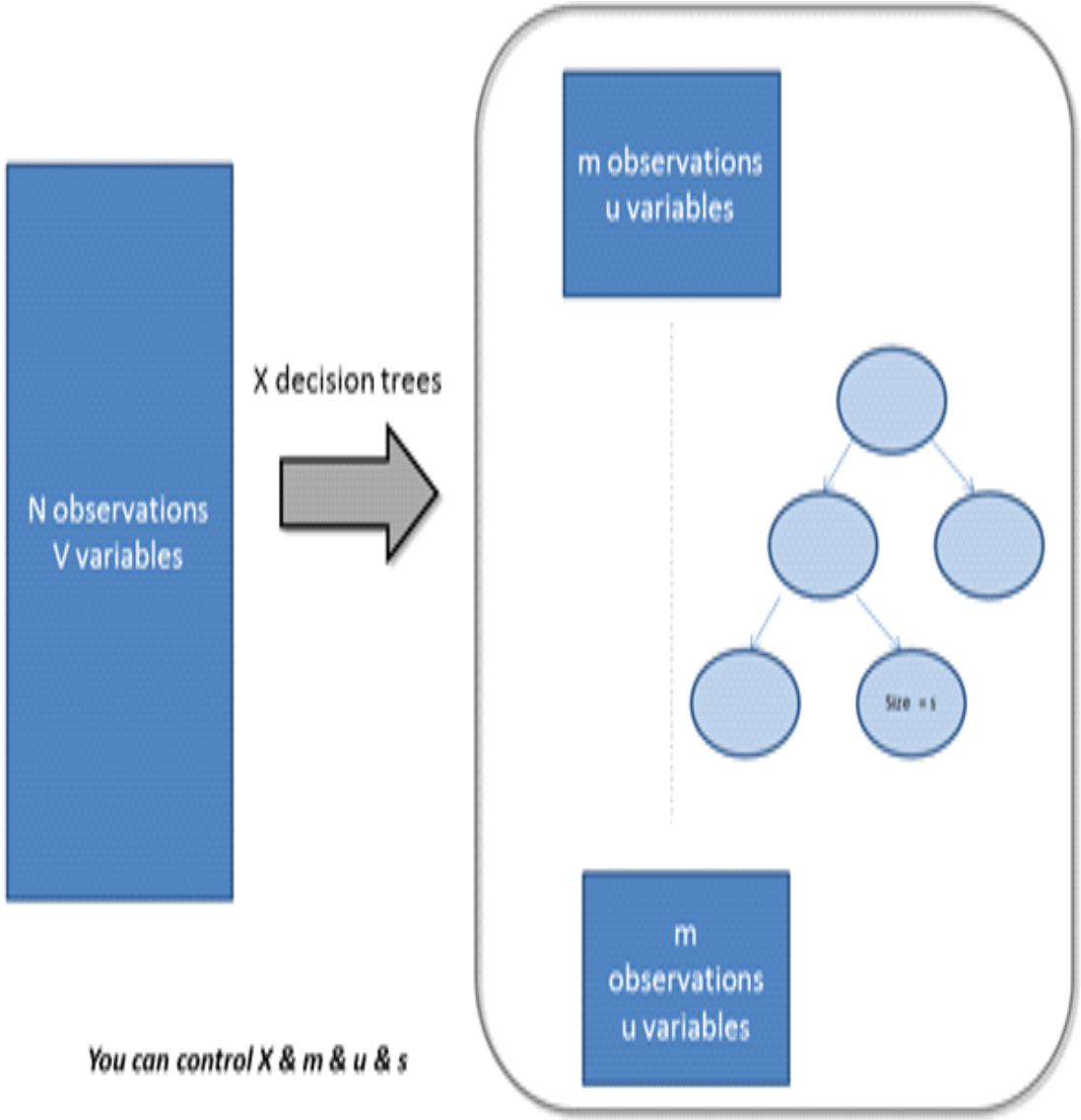
## Advantages

- Classification and Regression
- Variable importance plot
- Missing Values and Outliers
- Used for higher dimensions
- Get rid of over fitting***(Variance)

## Disadvantages

- Computation would be more
- Black box -No explainability /interpretability
- It surely does a great job for classification but not as for regression problems

# Parameters/Levers to tune RF:




**Max features:** MF is high, Speed down, strike a balance between high and low

**N Estimators:** no of decision trees ==> No of DT's increases ==> Speed down, complexity increases, result reliable

**Vertical depth**

**No of DN's/TN's**

**Min samples present at DN/TN**

**Random state: Set seed**

# Logistic Regression

- Logistic regression is one of the foundational tools for making classifications.
- The following example walks through basic logistic regression from start to finish.
- Let's say I wanted to examine the relationship between my basketball shooting accuracy and the distance that I shoot from.
- More specifically, I want a model that takes in "distance from the basket" in feet and spits out the probability that I will make the shot.
- First I need some data.
- So I went out and shot a basketball from various distances while recording each result (1 for a make, 0 for a miss). The result looks like this when plotted on a scatter plot:

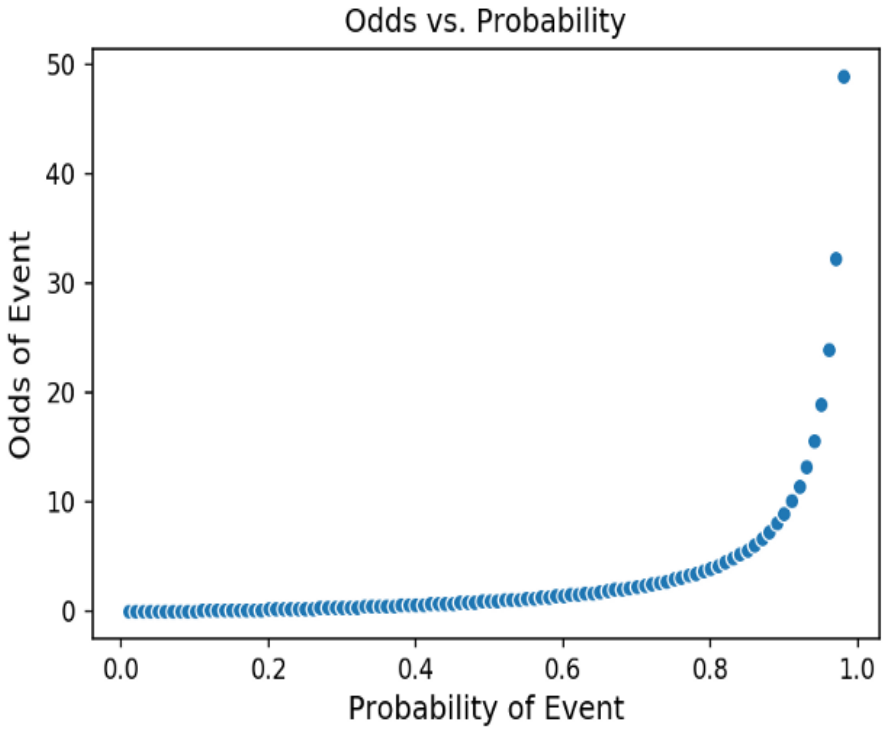
- Generally, the further I get from the basket, the less accurately I shoot.
- So we can already see the rough outlines of our model: when given a small distance, it should predict a high probability and when given a large distance it should predict a low probability.
- At a high level, logistic regression works a lot like good old linear regression.
- In linear regression, the output ***Y*** is in the same units as the target variable.
- However, in logistic regression the output ***Y*** is in log odds.
- Now unless you spend a lot of time sports betting or in casinos, you are probably not very familiar with odds.
- Odds is just another way of expressing the probability of an event, ***P(Event)***.

$$Odds = P(Event) / [1-P(Event)]$$

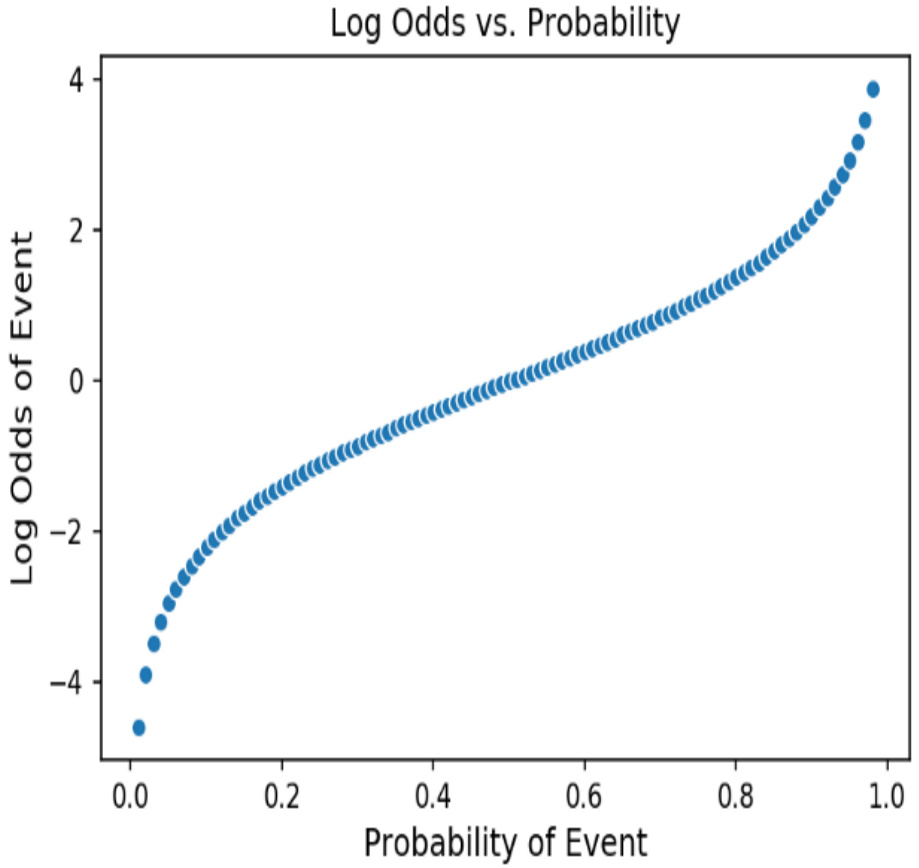
- Continuing our basketball theme, let's say I shot 100 free throws and made 70.
- Based on this sample, my probability of making a free throw is 70%. My odds of making a free throw can be calculated as:

$$\text{Odds} = 0.70 / (1–0.70) = 2.333$$

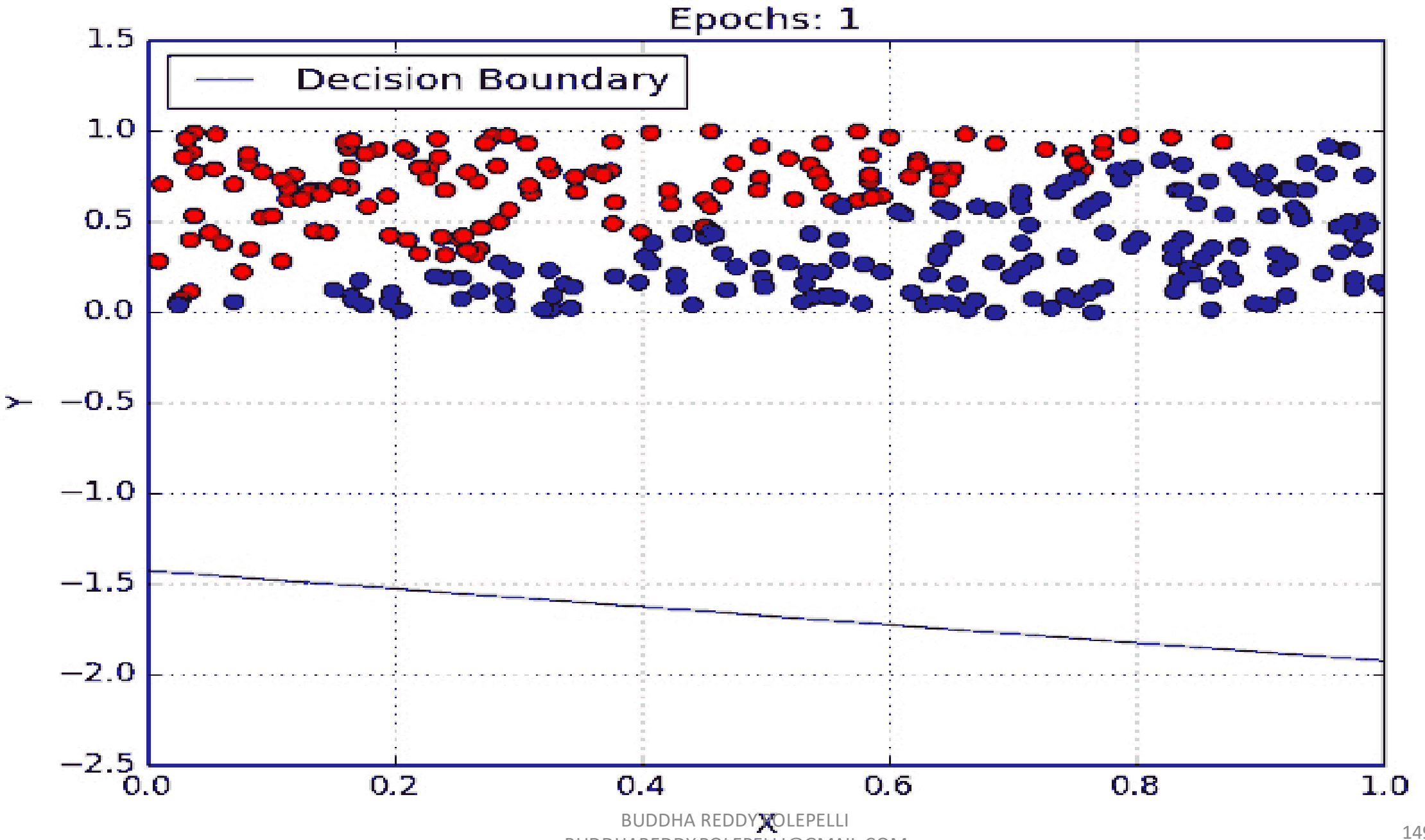
- Probabilities are bounded between 0 and 1, which becomes a problem in regression analysis.
- Odds as you can see below range from 0 to infinity.



- And if we take the natural log of the odds, then we get log odds which are unbounded (ranges from negative to positive infinity) and roughly linear across most probabilities
- Since we can estimate the log odds via logistic regression, we can estimate probability as well because log odds are just probability stated another way



- We can write our logistic regression equation:
- Z = B0 + B1*distance_from_basket
- where Z = log(odds_of_making_shot)
- And to get probability from Z, which is in log odds, we apply the sigmoid function.
- Applying the sigmoid function is a fancy way of describing the following transformation:
- Probability of making shot = 1 / [1 + e^(-Z)]

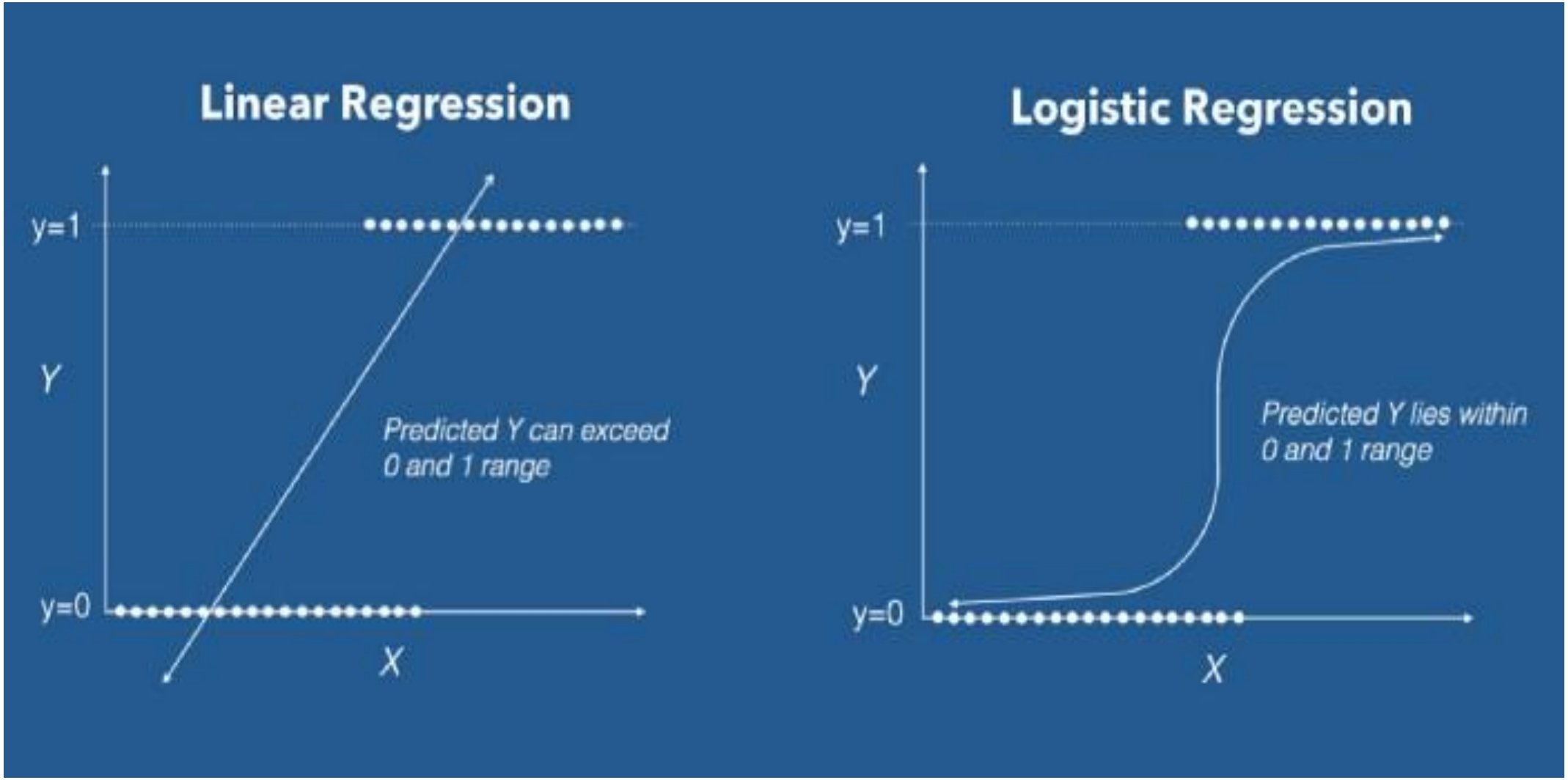

Epochs: 1
Decision Boundary
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
Y
0.0
0.2
0.4
0.6
0.8
1.0
X

# Logistic Regression

- Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability
- Logistic regression is used to assign observations to a discrete set of classes.
- Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign.
- Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

**What are the types of logistic regression**

- Binary (eg. Tumor Malignant or Benign)
- Multi-linear functions failsClass (eg. Cats, dogs or Sheep's)

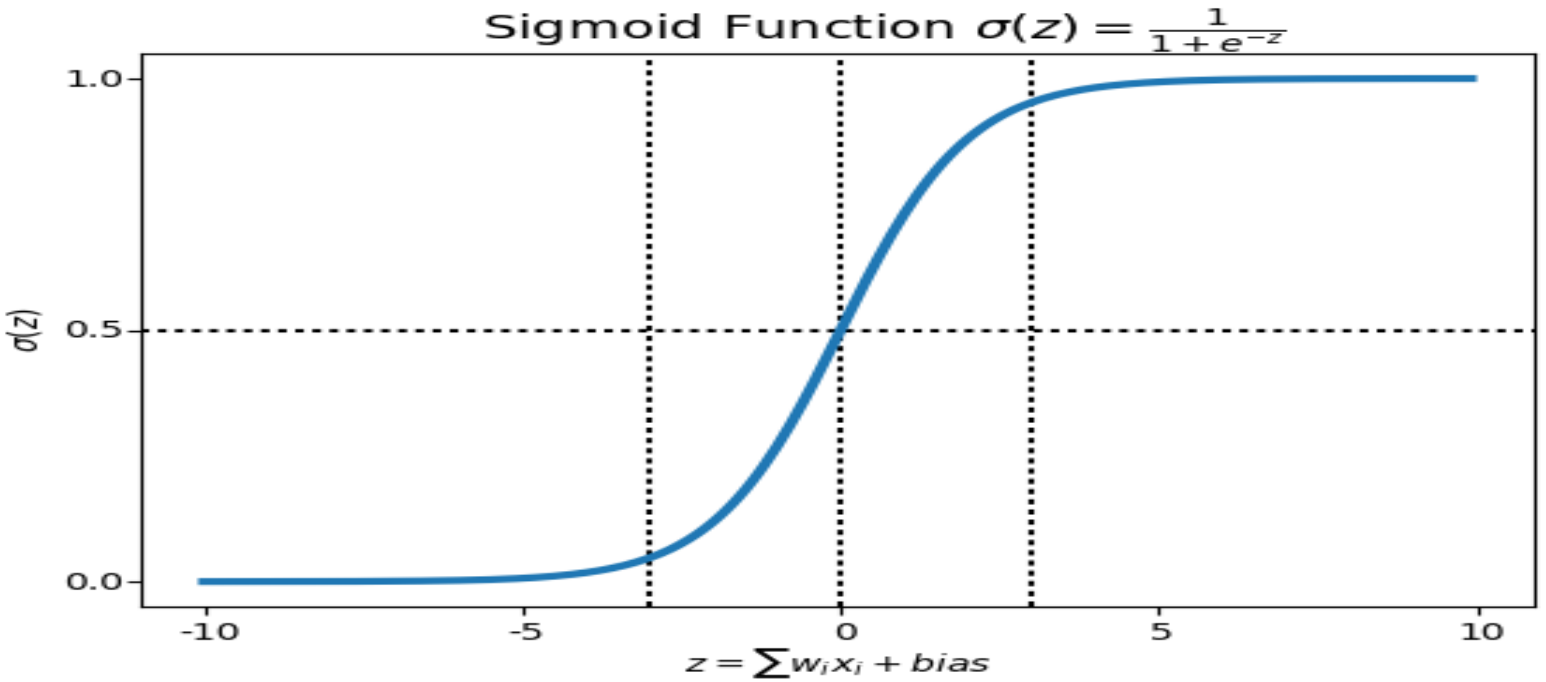

Linear Regression
y=1
Y
y=0
Predicted Y can exceed
0 and 1 range
X
Logistic Regression
y=1
Y
y=0
Predicted Y lies within
0 and 1 range
X

- We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the '**Sigmoid function**' or also known as the 'logistic function' instead of a linear function.
- The hypothesis of logistic regression tends it to limit the cost function between 0 and 1.
- Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$$0 \leq h_\theta(x) \leq 1$$

# What is the Sigmoid Function?

- In order to map predicted values to probabilities, we use the Sigmoid function.
- The function maps any real value into another value between 0 and 1.
- In machine learning, we use sigmoid to map predictions to probabilities.




$$f(x) = \frac{1}{1+e^{-(x)}}$$

# Hypothesis Representation

When using *linear regression* we used a formula of the hypothesis i.e.

$h\Theta(x) = \beta_0 + \beta_1 X$

For logistic regression we are going to modify it a little bit i.e.

$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$

We have expected that our hypothesis will give values between 0 and 1.

$Z = \beta_0 + \beta_1 X$

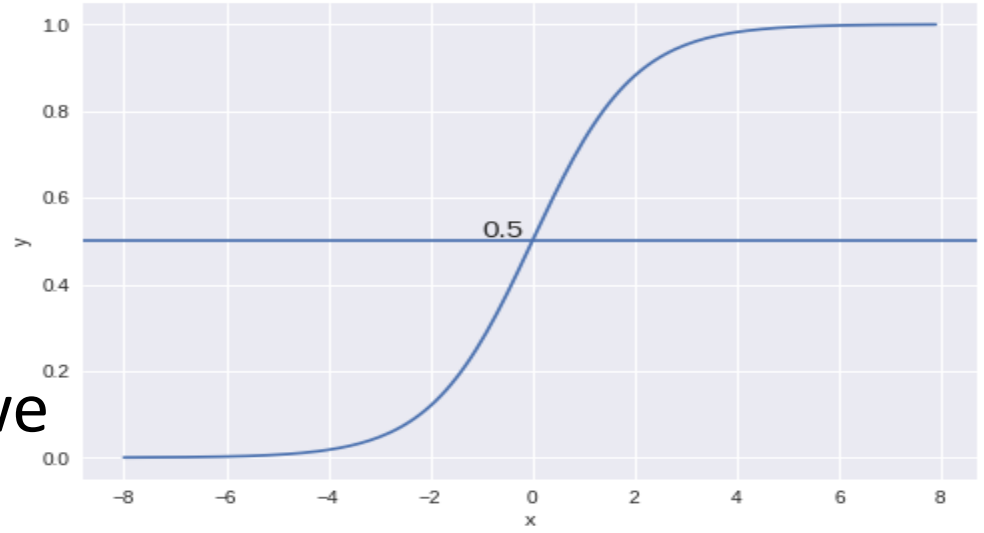
$h\Theta(x) = \text{sigmoid}(Z)$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$

- We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1.
- For Example, We have 2 classes, like cats and dogs(1 — dog , 0 — cats).
- We basically decide with a threshold value above which we classify values into Class 1 and of the value goes below the threshold then we classify it in Class 2.

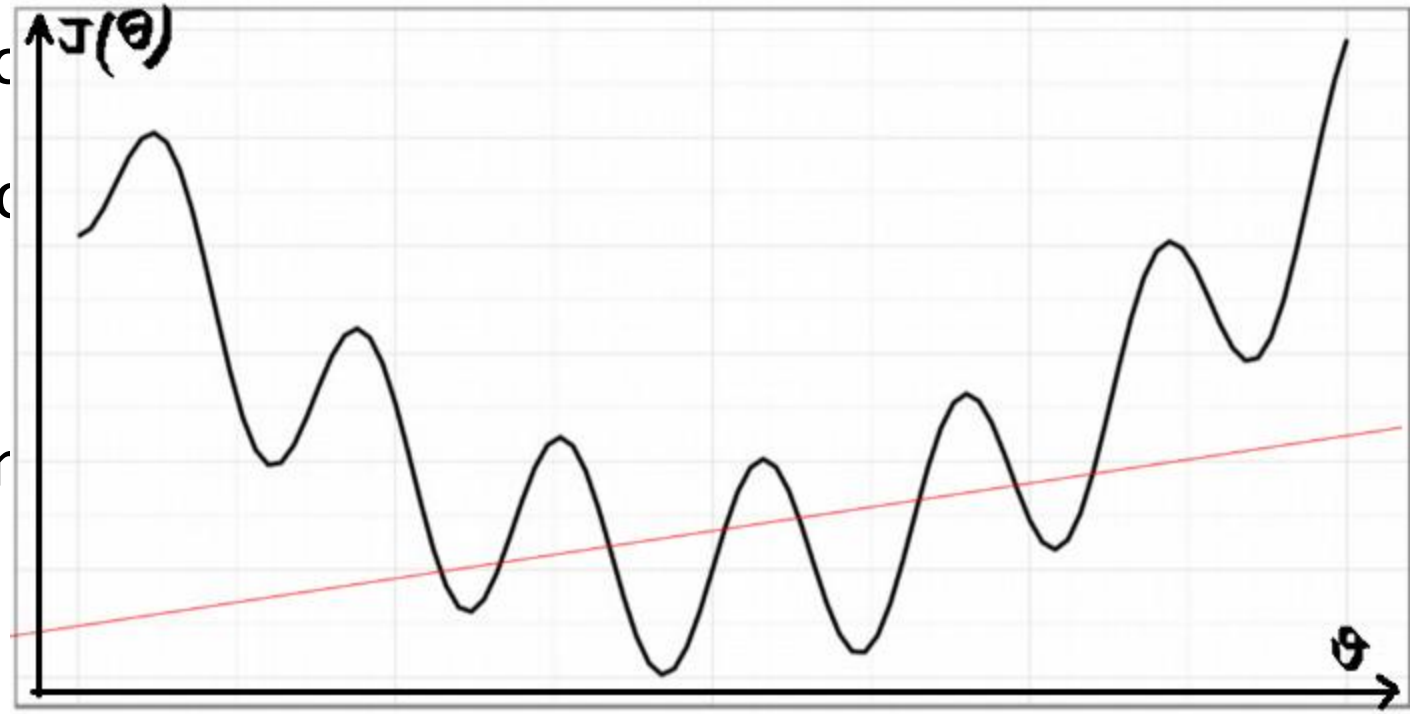
As shown in the graph we have chosen the threshold as 0.5, if the prediction function returned a value of 0.7 then we would classify this observation as Class 1(DOG).
If our prediction returned a value of 0.2 then we would classify the observation as Class 2(CAT).



- We learnt about the cost function $J(\vartheta)$ in the *Linear regression*, the cost function represents optimization objective i.e. we create a cost function and minimize it so that we can develop an accurate model with minimum error.

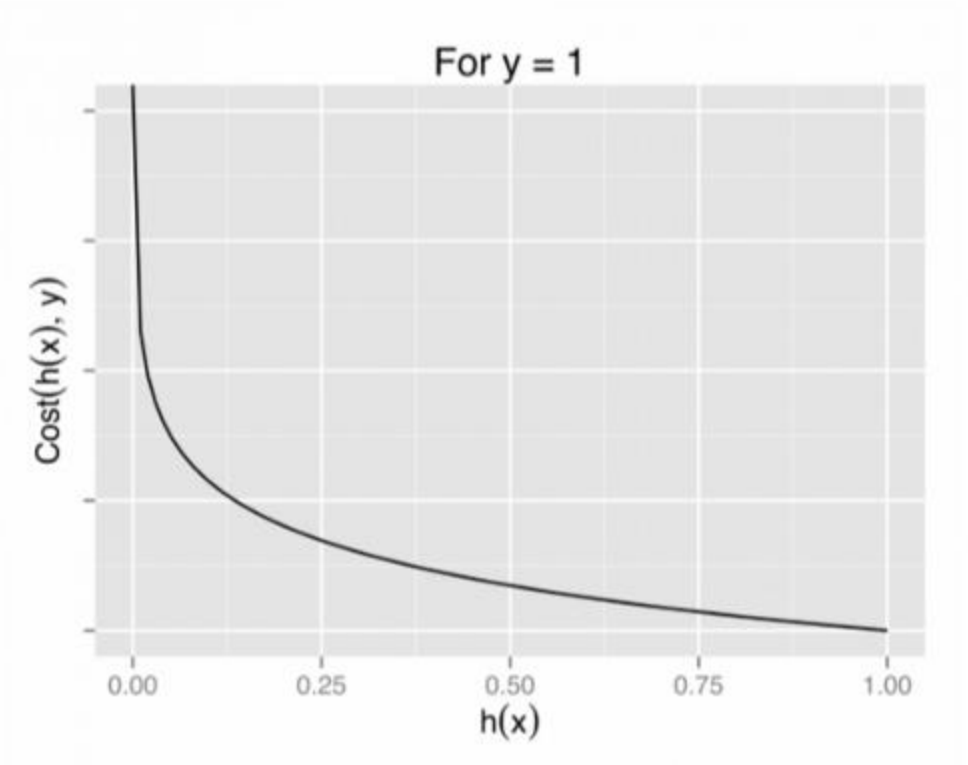
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

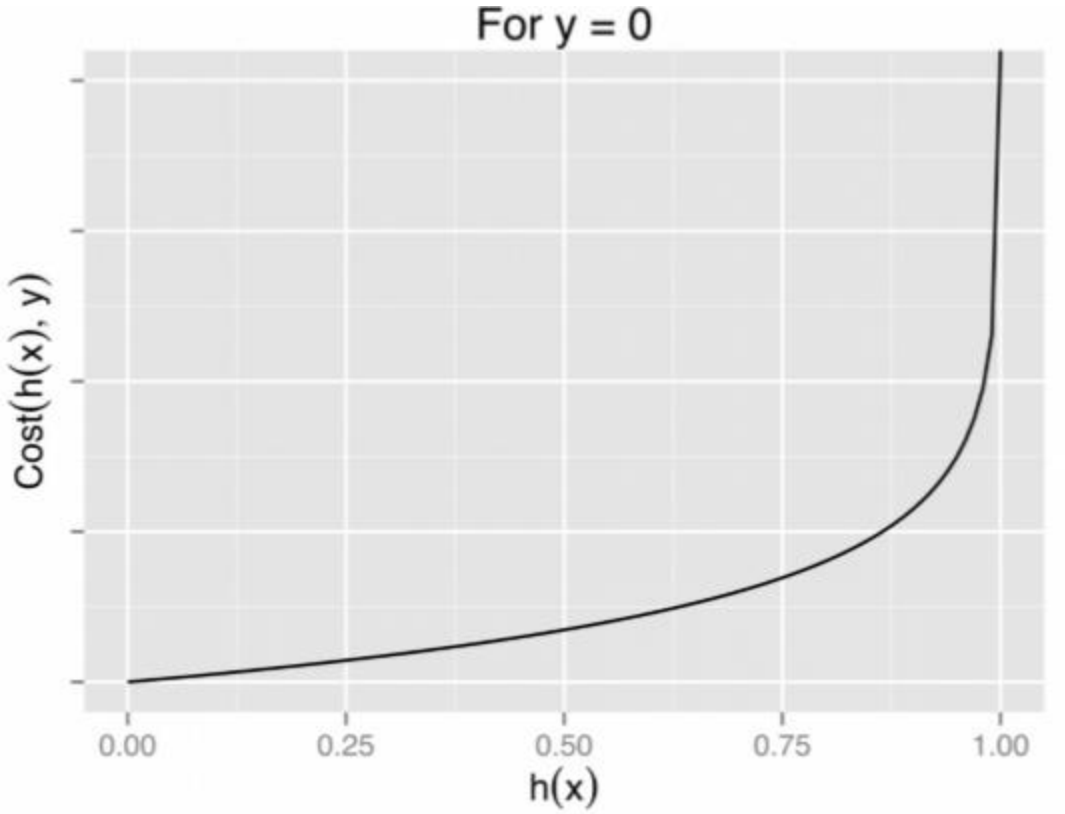
- If we try to use the cost function c
  then it would be of no use as it wo
  many local minimums, in which it
  **value** and find the global minimur



- For logistic regression, the Cost function is defined as:
- *–log(*hθ(x)*) if y = 1*
- *–log(1–*hθ(x)*) if y = 0*

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



- The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

## Gradient Descent

- Now the question arises, how do we reduce the cost value. Well, this can be done by using **Gradient Descent.**
- The main goal of Gradient descent is to **minimize the cost value.** i.e. min J(***ϑ***).
- Now to minimize our cost function we need to run the gradient descent function on each parameter i.e.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

- The assumptions made by logistic regression about the distribution and relationships in your data are much the same as the assumptions made in linear regression.
- **Binary Output Variable:** This might be obvious as we have already mentioned it, but logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.
- **Remove Noise:** Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.

- **Gaussian Distribution:** Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model. For example, you can use log, root, Box-Cox and other univariate transforms to better expose this relationship.
- **Remove Correlated Inputs:** Like linear regression, the model can overfit if you have multiple highly-correlated inputs. Consider calculating the pairwise correlations between all inputs and removing highly correlated inputs.
- **Fail to Converge:** It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

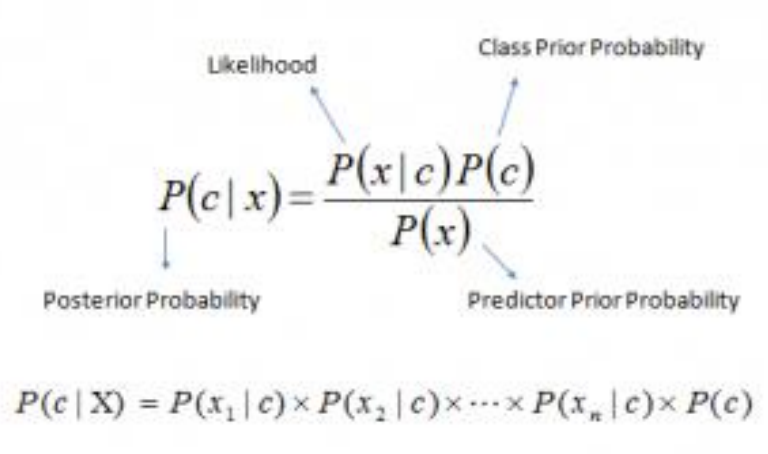
# Naïve Bayesian

# Naive Bayes Algorithm

- It is a classification technique based on Bayes' Theorem with an **assumption of independence among predictors. (**simple terms, assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
- Example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter.
- Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as '**Naive'.**
- Naive Bayes model is easy to build and **particularly useful for very large data sets.**
- Naive Bayes classifier is the fast, accurate and reliable algorithm

# Bayes Theorem

- In machine learning we are often interested in selecting the best hypothesis (h) given data (d).
- In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).
- One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.
- Bayes' Theorem is stated as: P(h|d) = (P(d|h) * P(h)) / P(d)
- Here P(h|d) is the probability of hypothesis h given the data d.
- P(d|h) is the probability of data d given that the hypothesis h was true.
- P(h) is the probability of hypothesis h being true (regardless of the data).
- P(d) is the probability of the data (regardless of the hypothesis).



# How Naive Bayes algorithm works?

- Naive Bayes classifier calculates the probability of an event in the following steps:
- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for each class
- Step 3: Put these value in Bayes Formula and calculate posterior probability.
- Step 4: See which class has a higher probability, given the input belongs to the higher probability class.
  - ( P(A|B) ) is the posterior probability of hypothesis ( A ) given the data ( B ).
  - ( P(B|A) ) is the likelihood of data ( B ) given that hypothesis ( A ) is true.
  - ( P(A) ) is the prior probability of hypothesis ( A ).
  - ( P(B) ) is the probability of the data ( B ).

# How Naive Bayes algorithm works?

| Email ID | Contains "Free" | Contains "Win" | Contains "Money" | Spam |
|---|---|---|---|---|
| 1 | Yes | No | Yes | Yes |
| 2 | No | Yes | No | No |
| 3 | Yes | Yes | Yes | Yes |
| 4 | No | No | No | No |
| 5 | Yes | No | No | No |

**Step 1: Calculate the prior probability for given class labels**

P(spam=yes) = 2/5= .4
p(spam=no)=3/5 = .6

**Step 2: Calculate Likelihoods**: For example, for the feature "Contains 'Free'": Repeat this for all features
p(free=yes/spam=yes)= 2/2
P(free=yes/spam=no)=1/3
P(win=yes/spam=yes)=1/2
P(win=yes/spam=no)=1/3
P(money=yes/spam=yes)=2/2
P(money=yes/spam=no) =0/3

**Step 3: posterior probabilites**:
P(features/spam=yes)=
p(free/spam)*p(win/spam)*p(money/spam)
=1.0*.5*1.0
p(features/spam=no)=p(free/no)*p(win/no)*
p(money/no)=
.33*.33*0=0

**Step 4: posterior probabilites**:
p(spam/features)=p(features/spam=yes)*p(spam=yes)
= .5*.4 =.2

P(nonspam/features)=p(features/spam=no)*p(spam=no)
=0*.6

# How Naive Bayes algorithm works?

- I have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing).
- Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.
- Step 1: Convert the data set into a frequency table

**First Approach (In case of a single feature)**

- Naive Bayes classifier calculates the probability of an event in the following steps:
- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for each class
- Step 3: Put these value in Bayes Formula and calculate posterior probability.
- Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

| Whether | Play |
|---|---|
| Sunny | No |
| Sunny | No |
| Overcast | Yes |
| Rainy | Yes |
| Rainy | Yes |
| Rainy | No |
| Overcast | Yes |
| Sunny | No |
| Sunny | Yes |
| Rainy | Yes |
| Sunny | Yes |
| Overcast | Yes |
| Overcast | Yes |
| Rainy | No |

**Frequency Table**

| Whether | No | Yes |
|---|---|---|
| Overcast | | 4 |
| Sunny | 2 | 3 |
| Rainy | 3 | 2 |
| Total | 5 | 9 |

**Likelihood Table 1**

| Whether | No | Yes | | |
|---|---|---|---|---|
| Overcast | | 4 | =4/14 | 0.29 |
| Sunny | 2 | 3 | =5/14 | 0.36 |
| Rainy | 3 | 2 | =5/14 | 0.36 |
| Total | 5 | 9 | | |
| | =5/14 | =9/14 | | |
| | 0.36 | 0.64 | | |

**Likelihood Table 2**

| Whether | No | Yes | Posterior Probability for No | Posterior Probability for Yes |
|---|---|---|---|---|
| Overcast | | 4 | 0/5=0 | 4/9=0.44 |
| Sunny | 2 | 3 | 2/5=0.4 | 3/9=0.33 |
| Rainy | 3 | 2 | 3/5=0.6 | 2/9=0.22 |
| Total | 5 | 9 | | |

**Probability of playing:**

*P(Yes|Overcast) = P(Overcast |Yes)* P(Yes) / P (Overcast)

**Calculate Prior Probabilities:**

P(Overcast) = 4/14 = 0.29

P(Yes)= 9/14 = 0.64

**Calculate Posterior Probabilities:**

P(Overcast |Yes) = 4/9 = 0.44

Put Prior and Posterior probabilities in equation

P (Yes | Overcast) = 0.44 * 0.64 / 0.29 = 0.98

***Probability of not playing:***

*P(No | Overcast) = P(Overcast | No) P(No) / P (Overcast)*

***Calculate Prior Probabilities:***

*P(Overcast) = 4/14 = 0.29*

*P(No)= 5/14 = 0.36*

**Calculate Posterior Probabilities:**

*P(Overcast |No) = 0/9 = 0*

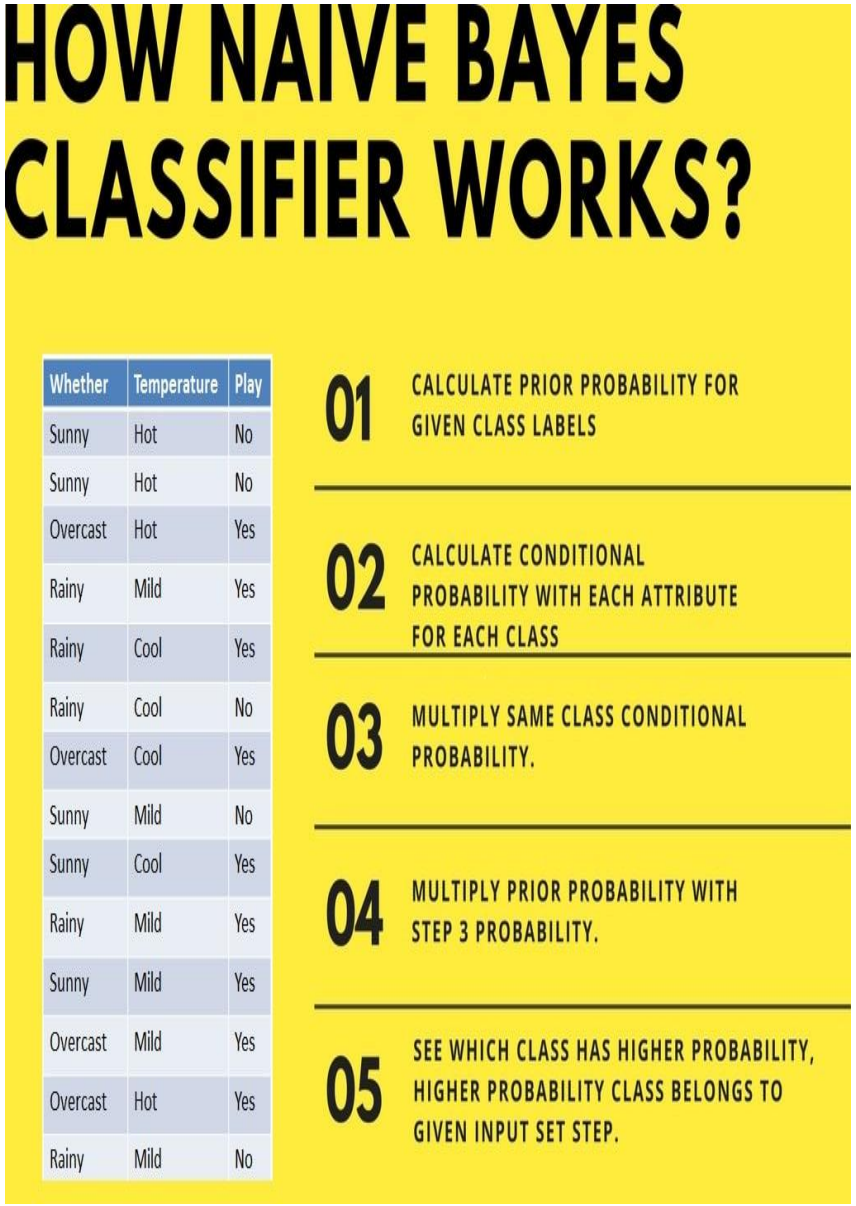
*Put Prior and Posterior probabilities in equation*

*P (No | Overcast) = 0 * 0.36 / 0.29 = 0*

*The probability of a 'Yes' class is higher.*

*So you can determine here if the weather is overcast than players will play the sport.*

## Second Approach (In case of multiple features)




Now suppose you want to calculate the probability of playing when the weather is overcast, and the temperature is mild.

**Probability of playing:**

*P(Play= Yes | Weather=Overcast, Temp=Mild) = P(Weather=Overcast, Temp=Mild | Play= Yes)*P(Play=Yes) .....(1)

*P(Weather=Overcast, Temp=Mild | Play= Yes)= P(Overcast |Yes)* P(Mild |Yes) ……..(2)

1.Calculate Prior Probabilities: P(Yes)= 9/14 = 0.64

2.Calculate Posterior Probabilities: P(Overcast |Yes) = 4/9 = 0.44 P(Mild |Yes) = 4/9 = 0.44

3.Put Posterior probabilities in equation (2) P(Weather=Overcast, Temp=Mild | Play= Yes) = 0.44 * 0.44 = 0.1936Put Prior and Posterior probabilities in equation (1)

4. P(Play= Yes | Weather=Overcast, Temp=Mild) = 0.1936*0.64 = 0.124

**Probability of not playing:**

*P(Play= No | Weather=Overcast, Temp=Mild) = P(Weather=Overcast, Temp=Mild | Play= No)*P(Play=No) .....(3)

*P(Weather=Overcast, Temp=Mild | Play= No)= P(Weather=Overcast |Play=No)* P(Temp=Mild | Play=No) .....(4)

1.Calculate Prior Probabilities: P(No)= 5/14 = 0.36

2.Calculate Posterior Probabilities: P(Weather=Overcast |Play=No) = 0/9 = 0 P(Temp=Mild | Play=No)=2/5=0.4

3.Put posterior probabilities in equation (4) P(Weather=Overcast, Temp=Mild | Play= No) = 0 *

# What are the Pros and Cons of Naive Bayes?

***Pros:***

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

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

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

# Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction.
- This is often known as “Zero Frequency”.
- To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

# Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

# How to build a basic model using Naive Bayes in Python

- scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under the scikit-learn library:
- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.
- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".
- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively

# Naive Bayes Classifier

- Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems.
- The technique is easiest to understand when described using binary or categorical input values.
- 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, i.e. that the attributes do not interact.
- Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

- The representation for naive Bayes is probabilities.
- A list of probabilities are stored to file for a learned naive Bayes model. This includes:
- **Class Probabilities**: The probabilities of each class in the training dataset.
- **Conditional Probabilities:** The conditional probabilities of each input value given each class value.
- Learning a naive Bayes model from your training data is fast.
- Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

# Calculating Class Probabilities

- The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances.
- For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:
- P(class=1) = count(class=1) / (count(class=0) + count(class=1))
- In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

# Calculating Conditional Probabilities

- The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value.
- For example, if a "*weather*" attribute had the values "*sunny*" and "*rainy*" and the class attribute had the class values "*go-out*" and "*stay-home*", then the conditional probabilities of each weather value for each class value could be calculated as:
- P(weather=sunny|class=go-out) = count(instances with weather=sunny and class=go-out) / count(instances with class=go-out)
- P(weather=sunny|class=stay-home) = count(instances with weather=sunny and class=stay-home) / count(instances with class=stay-home)
- P(weather=rainy|class=go-out) = count(instances with weather=rainy and class=go-out) / count(instances with class=go-out)
- P(weather=rainy|class=stay-home) = count(instances with weather=rainy and class=stay-home) / count(instances with class=stay-home)

# Make Predictions With a Naive Bayes Model

- Given a naive Bayes model, you can make predictions for new data using Bayes theorem.
- MAP(h) = max(P(d|h) * P(h))
- Using our example above, if we had a new instance with the *weather* of *sunny*, we can calculate:
- go-out = P(weather=sunny|class=go-out) * P(class=go-out)

  stay-home = P(weather=sunny|class=stay-home) * P(class=stay-home)
- We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:
- P(go-out|weather=sunny) = go-out / (go-out + stay-home)

  P(stay-home|weather=sunny) = stay-home / (go-out + stay-home)

- If we had more input variables we could extend the above example.
- For example, pretend we have a "*car*" attribute with the values "*working*" and "*broken*". We can multiply this probability into the equation.
- For example below is the calculation for the "go-out" class label with the addition of the car input variable set to "working":
- go-out = P(weather=sunny|class=go-out) * P(car=working|class=go-out) * P(class=go-out).
- **Gaussian Naive Bayes**
- Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution.
- This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

## Representation for Gaussian Naive Bayes

- Above, we calculated the probabilities for input values for each class using a frequency. With real-valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution.
- This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

## Learn a Gaussian Naive Bayes Model From Data

- This is as simple as calculating the mean and standard deviation values of each input variable (x) for each class value.
  mean(x) = 1/n * sum(x)
- Where n is the number of instances and x are the values for an input variable in your training data.
- We can calculate the standard deviation using the following equation: standard deviation(x) = sqrt(1/n * sum(xi-mean(x)^2 ))
- This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, sqrt() is the square root function, sum() is the sum function, xi is a specific value of the x variable for the i'th instance and mean(x) is described above, and ^2 is the square.

## Make Predictions With a Gaussian Naive Bayes Model

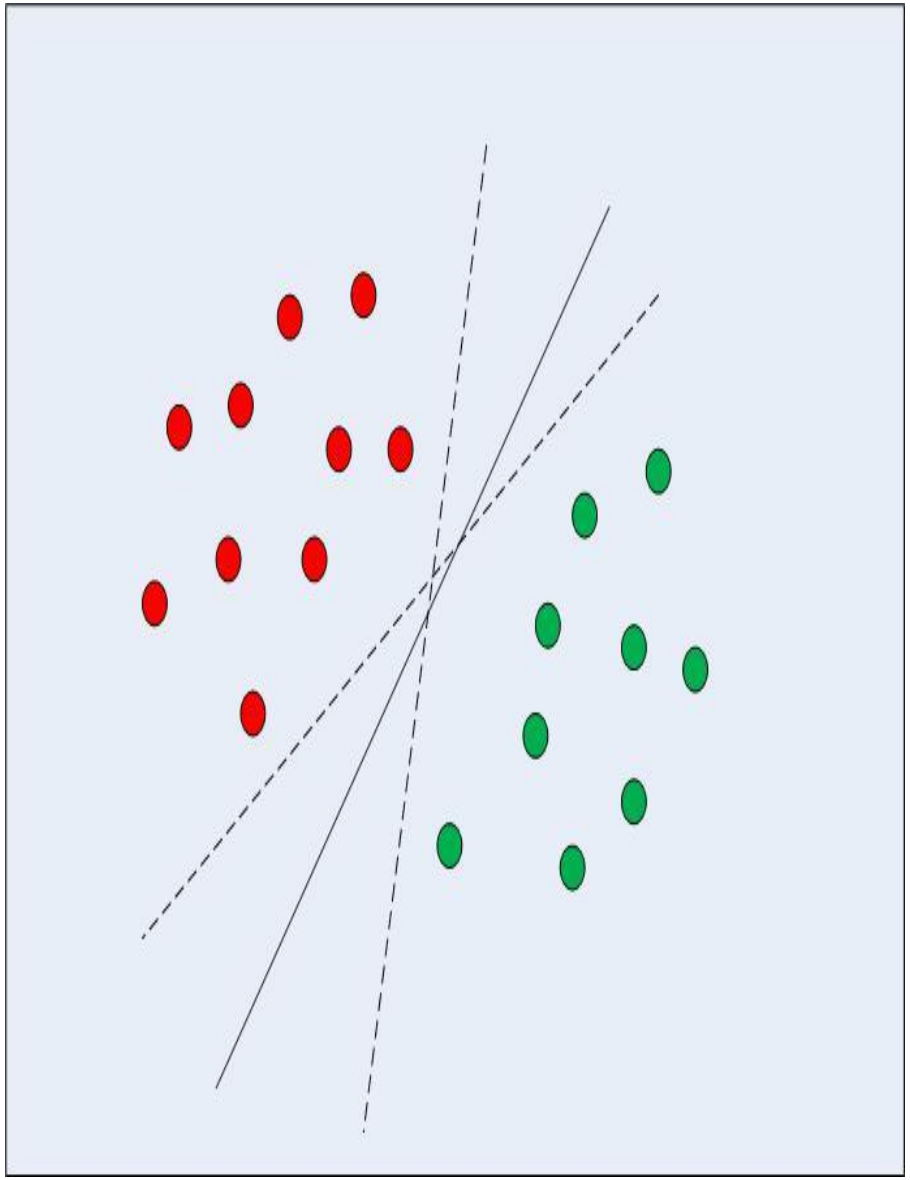
- Probabilities of new x values are calculated using the Gaussian Probability Density Function (PDF).
- When making predictions these parameters can be plugged into the Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value for that class. pdf(x, mean, sd) = (1 / (sqrt(2 * PI) * sd)) * exp(-((x-mean^2)/(2*sd^2)))
- Where pdf(x) is the Gaussian PDF, sqrt() is the square root, mean and sd are the mean and standard deviation calculated above, PI is the numerical constant, exp() is the numerical constant e or Euler's number raised to power and x is the input value for the input variable.
- We can then plug in the probabilities into the equation above to make predictions with real-valued inputs.
- For example, adapting one of the above calculations with numerical values for weather and car:
- go-out = P(pdf(weather)|class=go-out) * P(pdf(car)|class=go-out) * P(class=go-out)

# Best Prepare Your Data For Naive Bayes

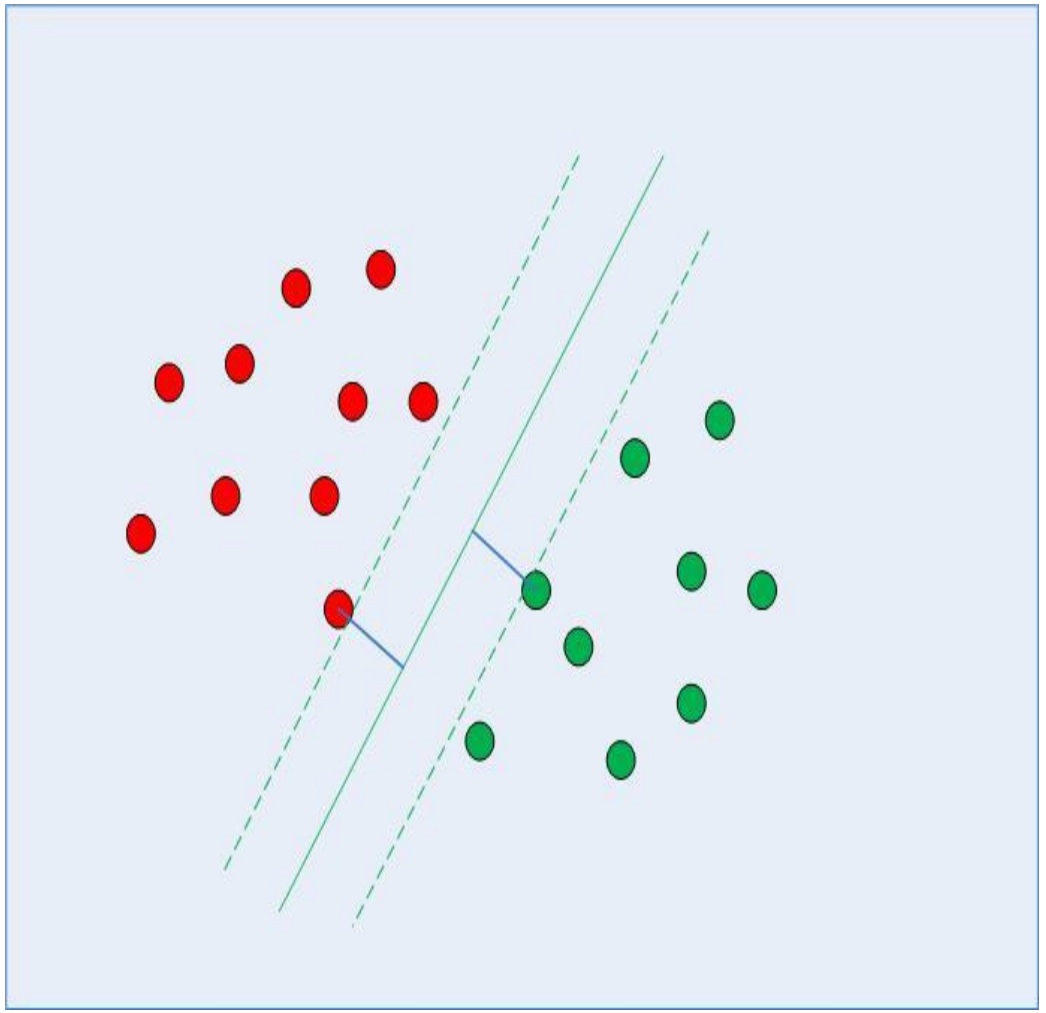
- **Categorical Inputs**: Naive Bayes assumes label attributes such as binary, categorical or nominal.
- **Gaussian Inputs**: If the input variables are real-valued, a Gaussian distribution is assumed. In which case the algorithm will perform better if the univariate distributions of your data are Gaussian or near-Gaussian. This may require removing outliers (e.g. values that are more than 3 or 4 standard deviations from the mean).
- **Classification Problems**: Naive Bayes is a classification algorithm suitable for binary and multiclass classification.
- **Log Probabilities**: The calculation of the likelihood of different class values involves multiplying a lot of small numbers together. This can lead to an underflow of numerical precision. As such it is good practice to use a log transform of the probabilities to avoid this underflow.
- **Kernel Functions**: Rather than assuming a Gaussian distribution for numerical input values, more complex distributions can be used such as a variety of kernel density functions.
- **Update Probabilities**: When new data becomes available, you can simply update the probabilities of your model. This can be helpful if the data changes frequently.

# SVM

- Simple SVM algorithm can be used to find decision boundary for linearly separable data (in two dimensions).
- A typical machine learning algorithm tries to find a boundary that divides the data in such a way that the misclassification error can be minimized.
- If you closely look figure there can be several boundaries that correctly divide the data points. The two dashed lines as well as one solid line classify the data correctly.
- SVM differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes.
- An SVM doesn't merely find a decision boundary; it finds the most optimal decision boundary.
- The most optimal decision boundary is the one which has maximum margin from the nearest points of all the classes

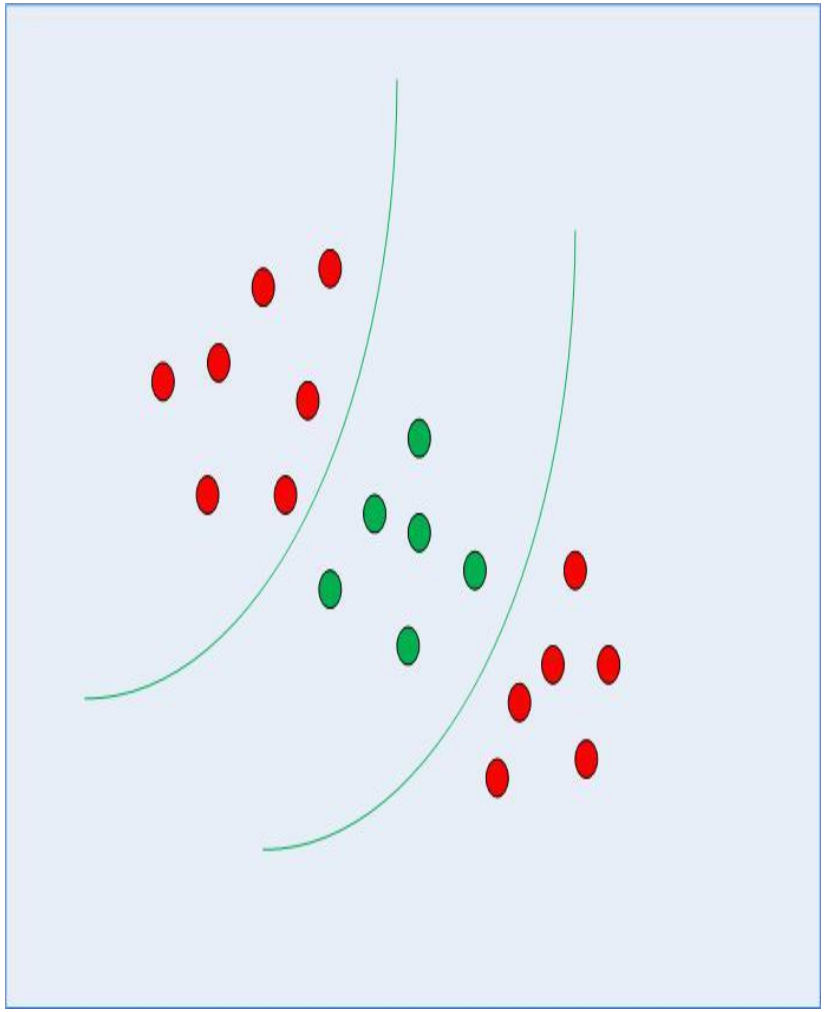


- The nearest points from the decision boundary that maximize the distance between the decision boundary and the points are called support vectors as seen in figure
- The decision boundary in case of support vector machines is called the maximum margin classifier, or the maximum margin hyper plane.
- There is complex mathematics involved behind finding the support vectors, calculating the margin between decision boundary and the support vectors and maximizing this margin. In this tutorial we will not go into the detail of the mathematics, we will rather see how SVM and Kernel SVM are implemented via the Python Scikit-Learn library.



# Kernel SVM

- in the case of non-linearly separable data, such as the one shown in Figure, a straight line cannot be used as a decision boundary.
- In case of non-linearly separable data, the simple SVM algorithm cannot be used.
- Rather, a modified version of SVM, called Kernel SVM, is used.
- Basically, the kernel SVM projects the non-linearly separable data lower dimensions to linearly separable data in higher dimensions in such a way that data points belonging to different classes are allocated to different dimensions.



# Unsupervised machine learning

# Clustering

# What is clustering

- It is the process of grouping similar entities together
- The goal of clustering is to find similarities in the data points and group similar data points together
- It is used for knowledge discovery rather than prediction.
- It provides an insight into the natural groupings found within data.

# Applications:

- Segmenting customers into groups with similar demographics or buying patterns for targeted campaigns
- Detecting anomalous behavior such as unauthorized intrusions into computer networks by identifying the patterns of use falling outside of the groups/clusters

# Why do we need clustering:

- It is a technique generally used to do initial profiling of the portfolio.
- After having a good understanding of the portfolio, an objective modeling technique is used to build specific strategy

# Hierarchical Clustering:

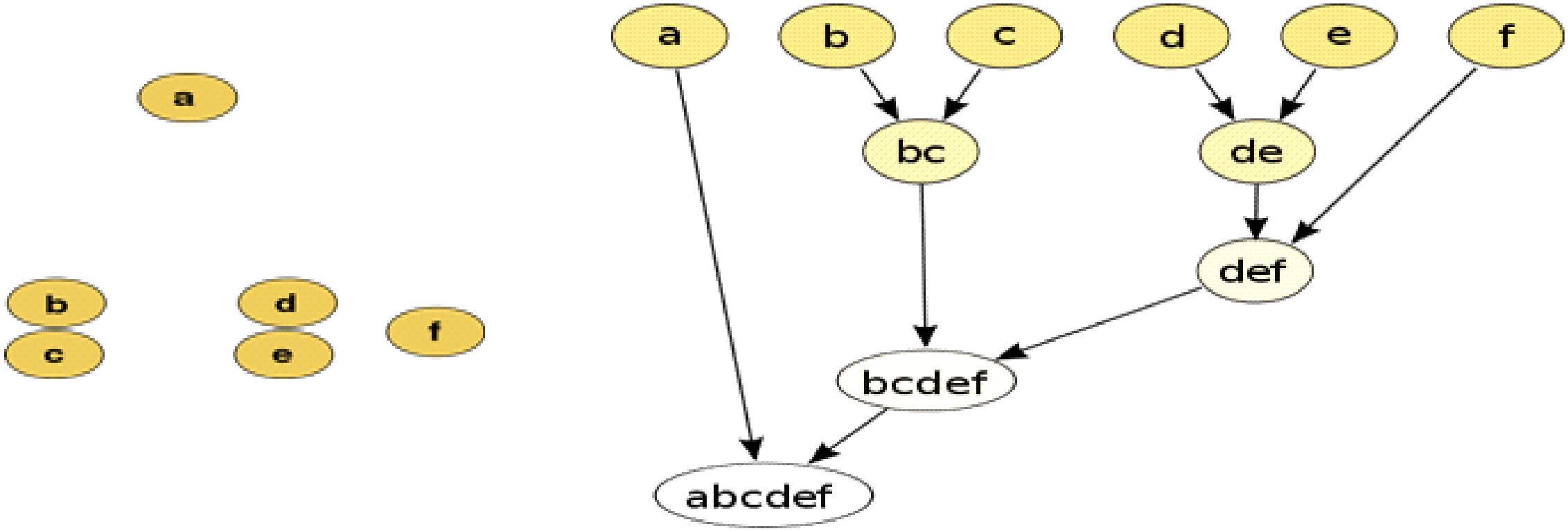
- HC starts with all the data points assigned to a cluster of its own
- In each iterations, we compare each data cluster against the other and then two nearest clusters are merged into the same cluster
- We continue this process and this algorithm terminates when there is only one single cluster.

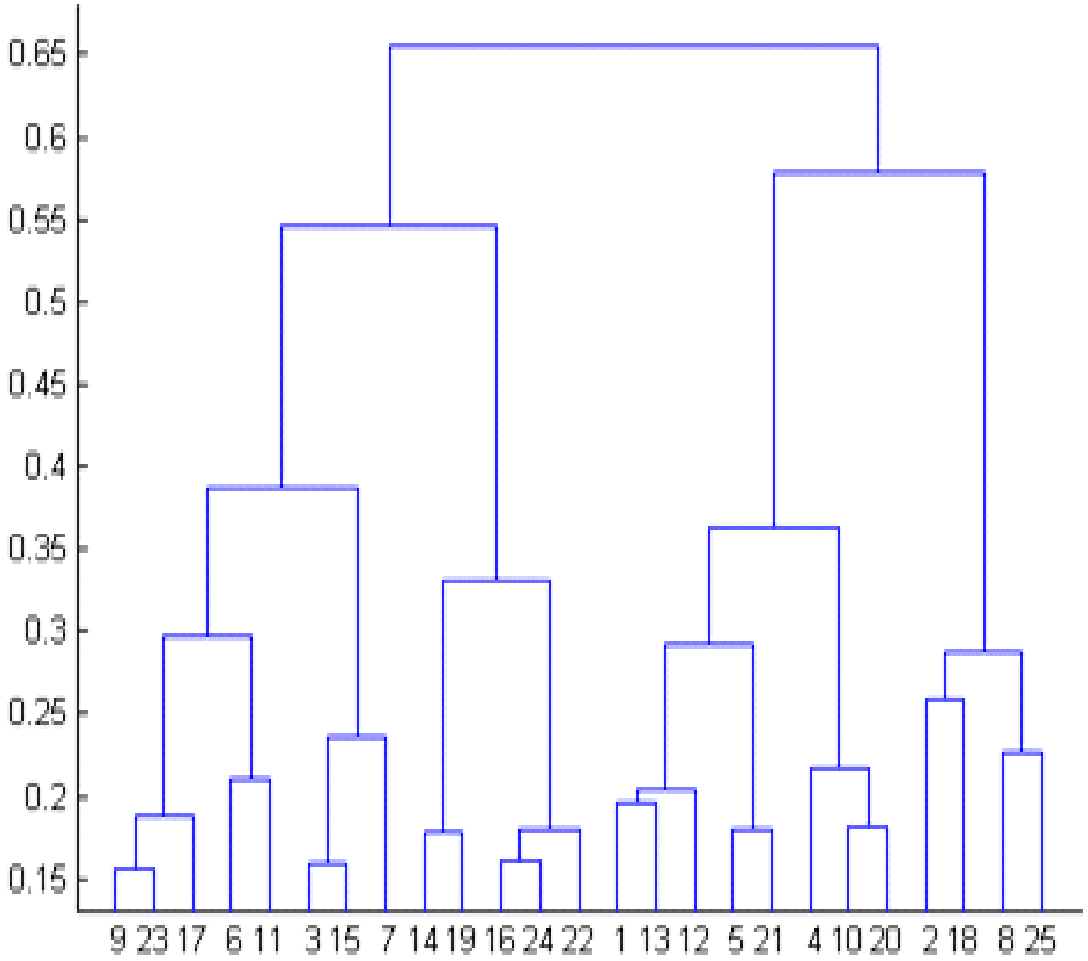
**Note:**

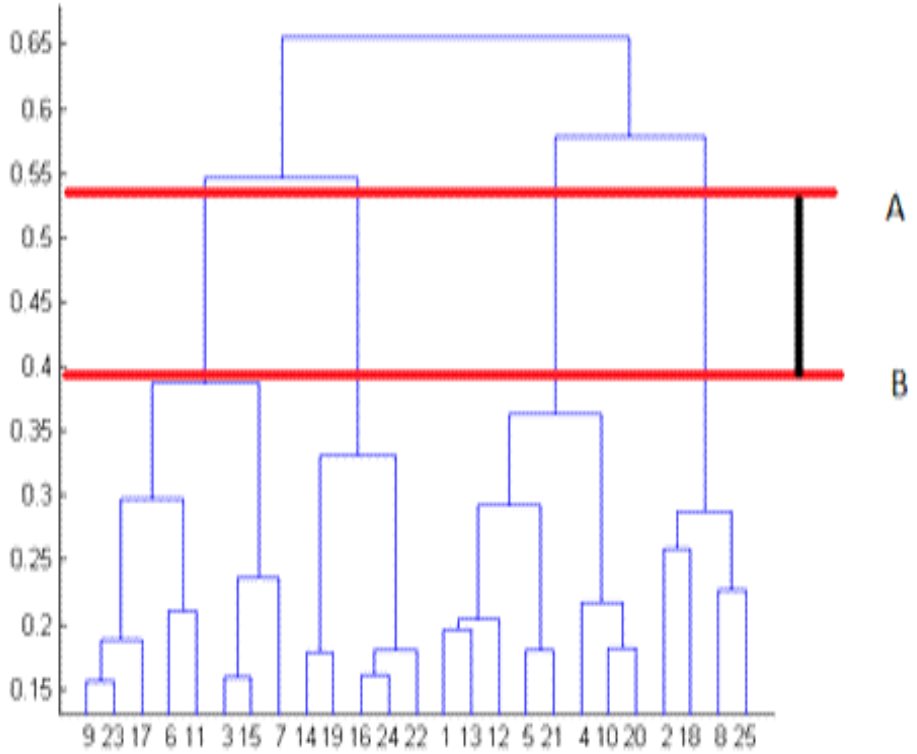
- The only problem with this technique that it is able to handle only small number of data points and is very time consuming. Because it tries to calculate the distance between all the possible combinations and then takes one decision to combine two groups/individual data points

# Hierarchical Clustering:

- This technique operates on the simplest principle, a data point closer to the base point will behave more similar compared to a data point which is far from base point.




0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
9 23 17 6 11 3 15 7 14 19 16 24 22 1 13 12 5 21 4 10 20 2 18 8 25

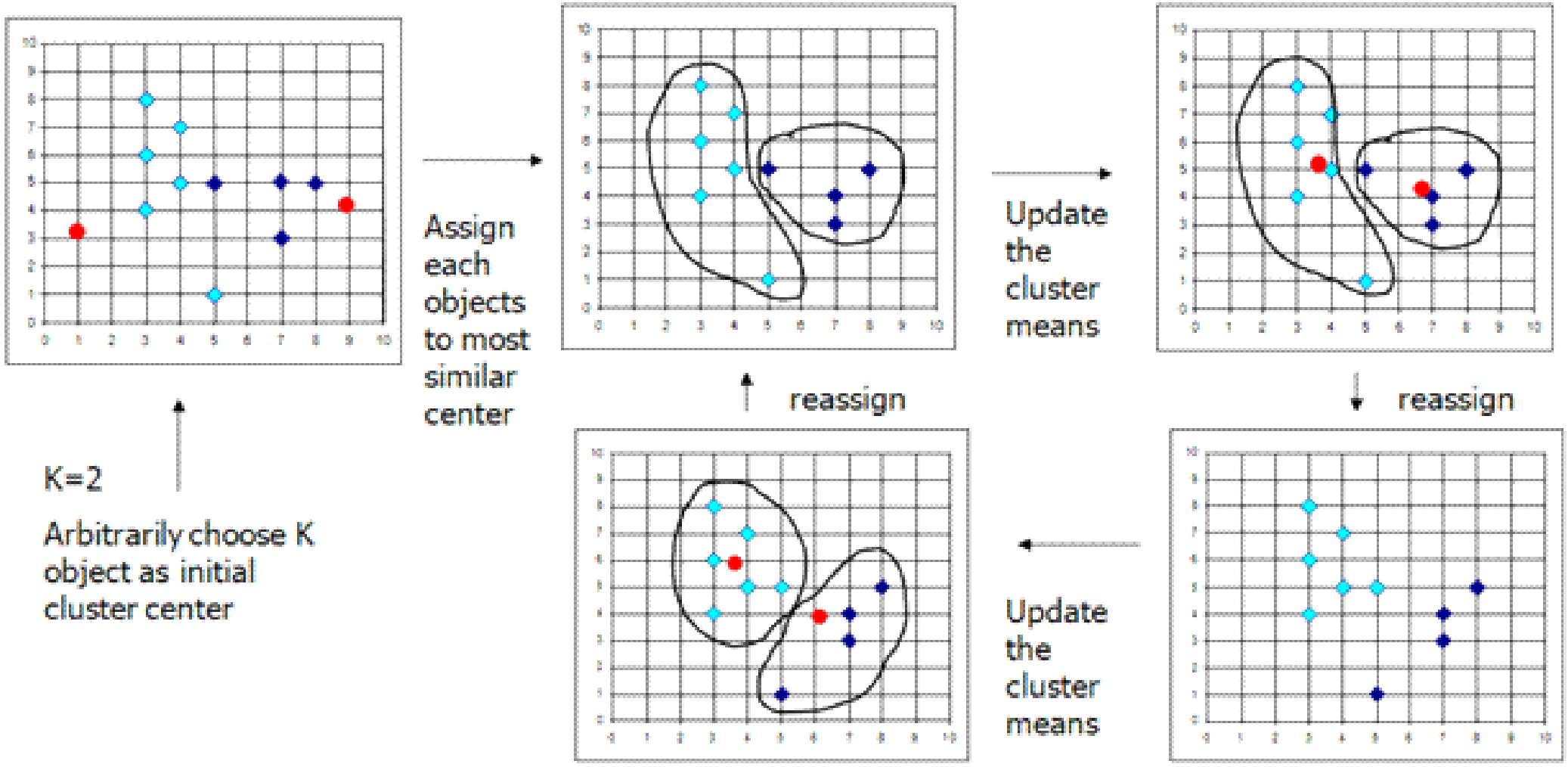

0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
A
B
9 23 17 6 11 3 15 7 14 19 16 24 22 1 13 12 5 21 4 10 20 2 18 8 25

- The best choice of the number of clusters is "the number of verticals lines in the dendogram cut by a horizontal line that can traverse the maximum distance vertically without intersecting a cluster“.
- This algorithm has been implemented using bottom up approach
- It is also possible to follow top-down starting with all data points assigned in the same cluster and recursively split till each data point is assigned to a separate cluster

**The decision of merging clusters is taken on basis of closeness of these clusters**

- Euclidean distance*
- Manhattan distance
- Hamming
- Jacard
- Mahalanobis
- Cosine (*text)

# K Means Clustering

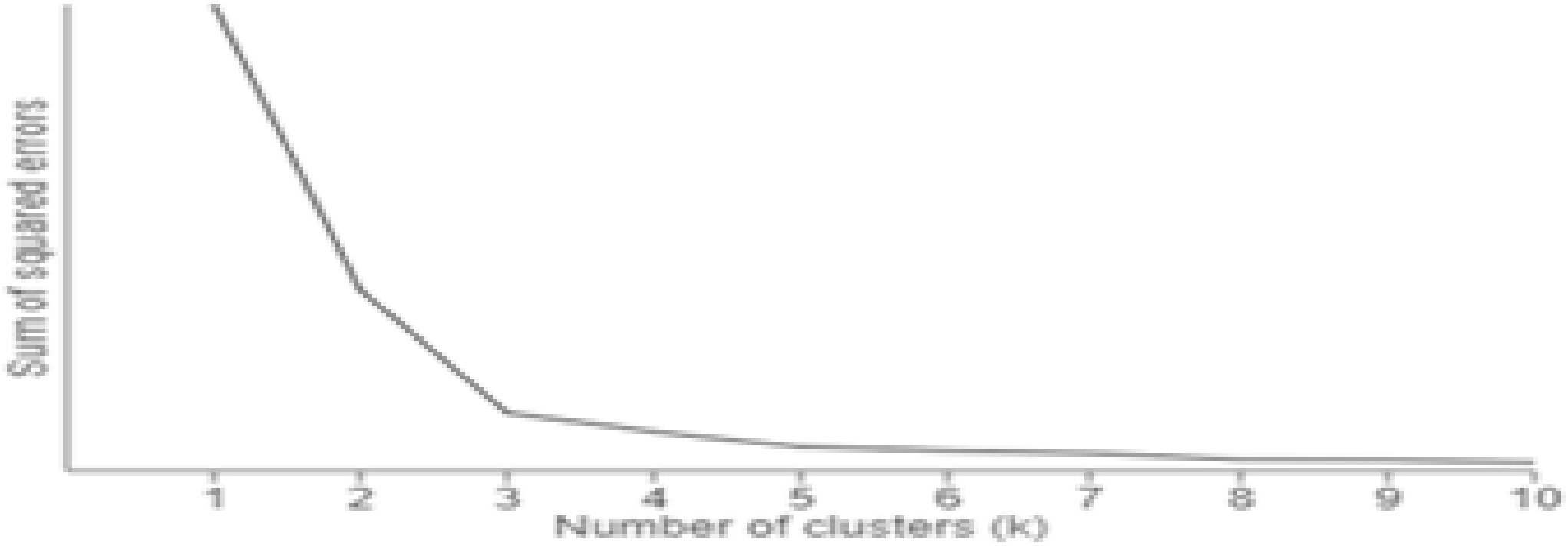


# steps

- Specify the desired number of clusters 'k'
- Randomly assign each data point to the nearest cluster center
- Compute cluster centroids
- Reassign each data point to the closest cluster centroid
- Re-compute cluster centroids
- Repeat the above steps until no improvements are possible

# Decide value of K

- Using elbow method*
- K means clustering on a range of k values and plot the "percentage of variance explained" on Y axis and "k values" on X axis



# Definition of a good cluster analysis:

- **Data points within same cluster share similar profile:** Within variance should be less and across variance should be high. one SD deviation distance between the clusters
- **Well spread proportion of data points among clusters:** ~min of 5% and max of 35%

# Difference between 'K' Means and Hierarchical Clustering:

- KC works on large datasets as against HC
- In HC, number of clusters are decided after the model building whereas in KC, we have to decide the k value prior to build
- Results are reproducible in HC and not for KC


ASKING THE RIGHT
QUESTIONS
TAKES AS MUCH SKILL
AS GIVING THE RIGHT
ANSWERS."
- ROBERT HALF