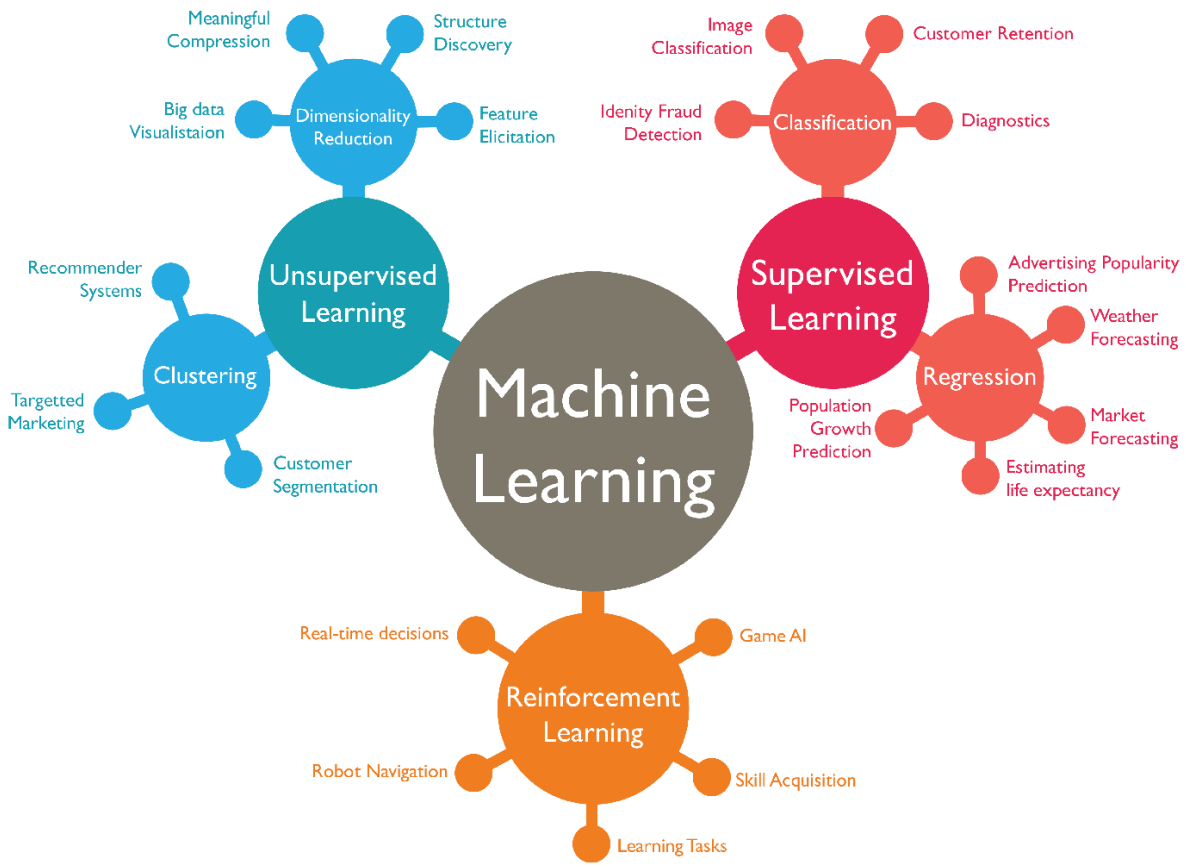
# Machine Learning

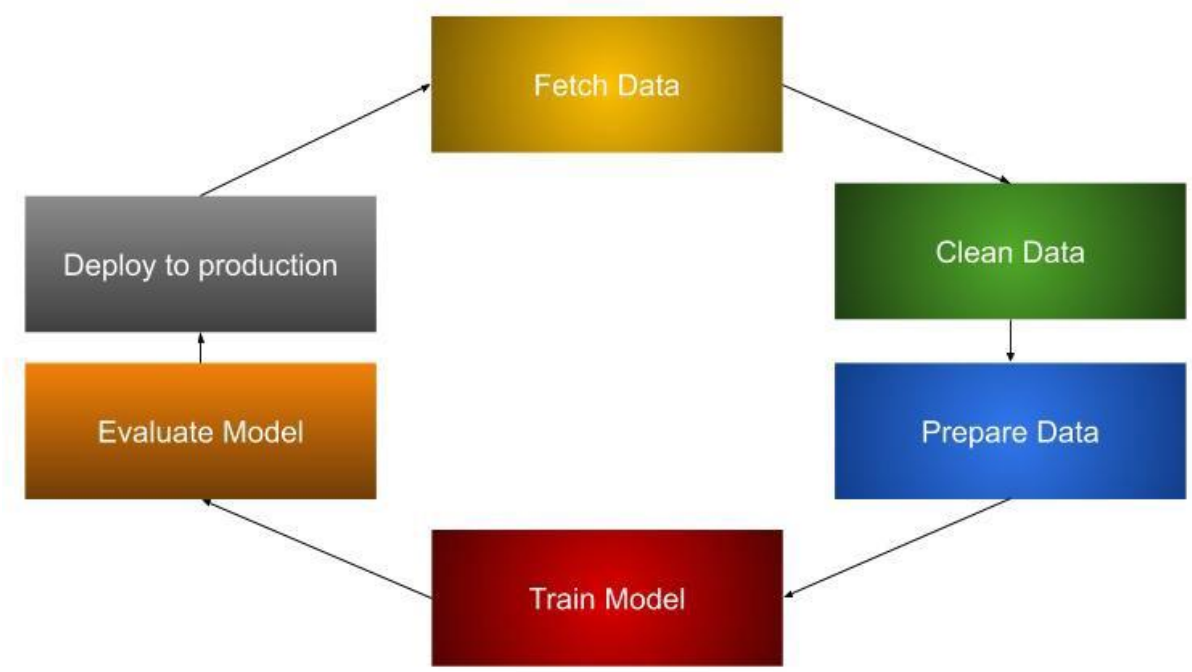
Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data.

These are three types of machine learning:

- supervised learning.
- unsupervised learning.
- reinforcement learning.



# Project Flow




# Exploratory Data Analysis

Exploratory data analysis is the first and foremost step to analyse any kind of data. Rather than a specific set of procedures, EDA is an approach, or a philosophy, which seeks to explore the most important and often hidden patterns in a data set. In EDA, we explore the data and try to come up with a hypothesis about it which we can later test using hypothesis testing. Statisticians use it to take a bird's eye view of the data and try to make some sense of it.

EDA has five steps:

- Data sourcing
- Data cleaning
- Univariate analysis
- Bivariate analysis
- Derived metrics

## Data Sourcing:

To solve a business problem using analytics, you need to have historical data to come up with actionable insights. Data is the key — the better the data, the more insights you can get out of it.

## Data Cleaning:

There are various types of quality issues when it comes to data, and that's why data cleaning is one of the most time-consuming steps of data analysis.

- Fix rows and columns
- Fix missing values
- Standardise values
- Fix invalid values
- Filter data

### Fix rows and columns:

**Checklist for Fixing Rows:**

- Delete summary rows: Total, Subtotal rows
- Delete incorrect rows: Header rows, Footer rows
- Delete extra rows: Column number, indicators, Blank rows, Page No.

**Checklist for Fixing Columns**

- Merge columns for creating unique identifiers if needed: E.g. Merge State, City into Full address
- Split columns for more data: Split address to get State and City to analyse each separately
- Add column names: Add column names if missing
- Rename columns consistently: Abbreviations, encoded columns
- Delete columns: Delete unnecessary columns
- Align misaligned columns: Dataset may have shifted columns

### Fix Missing values:

- **Set values as missing values**: Identify values that indicate missing data, and yet are not recognised by the software as such, e.g treat blank strings, "NA", "XX", "999", etc. as missing.
- **Adding is good, exaggerating is bad**: You should try to get information from reliable external sources as much as possible, but if you can't, then it is better to keep missing values as such rather than exaggerating the existing rows/columns.
- **Delete rows, columns**: Rows could be deleted if the number of missing values are insignificant in number, as this would not impact the analysis. Columns could be removed if the missing values are quite significant in number.
- **Fill partial missing values using business judgement**: Missing time zone, century, etc. These values are easily identifiable.

### Standardising values:

- **Standardise units**: Ensure all observations under a variable have a common and consistent unit, e.g. convert lbs to kgs, miles/hr to km/hr, etc.
- **Scale values if required**: Make sure the observations under a variable have a common scale
- **Standardise precision** for better presentation of data, e.g. 4.5312341 kgs to 4.53 kgs.
- **Remove outliers**: Remove high and low values that would disproportionately affect the results of your analysis.

### Filtering Data:

- **Deduplicate data**: Remove identical rows, remove rows where some columns are identical
- **Filter rows**: Filter by segment, filter by date period to get only the rows relevant to the analysis
- **Filter columns**: Pick columns relevant to the analysis
- **Aggregate data**: Group by required keys, aggregate the rest

## Univariate Analysis:

As the term "**univariate**" suggests, this session deals with analysing variables one at a time. It is important to separately understand each variable before moving on to analysing multiple variables together.

## Bivariate Analysis:

Correlation is a metric to find the relationship between the variables. It is a number between -1 and 1 which quantifies the extent to which two variables 'correlate' with each other.

- If one increases as the other increases, the correlation is positive
- If one decreases as the other increases, the correlation is negative
- If one stays constant as the other varies, the correlation is zero

**Correlation Matrix**

| | Var1 | Var2 | Var3 | Var4 | Var5 |
|---|---|---|---|---|---|
| Var1 | 1 | -0.08071 | 0.098675 | 0.014625 | 0.061913 |
| Var2 | -0.08071 | 1 | -0.10168 | 0.37678 | 0.103062 |
| Var3 | 0.098675 | -0.10168 | 1 | 0.049934 | 0.119171 |
| Var4 | 0.014625 | 0.37678 | 0.049934 | 1 | 0.002249 |
| Var5 | 0.061913 | 0.103062 | 0.119171 | 0.002249 | 1 |

## Derived Metrices:

Process of deriving a new column out of existing column, which helps in simplified form of data in data analysis.

# Central Limit Theorem

# Hypothesis testing

**Hypothesis testing** is used to confirm your conclusion (or hypothesis) about the population parameter (which you know from EDA or your intuition). Through hypothesis testing, you can determine whether there is enough evidence to conclude if the hypothesis about the population parameter is true or not.

Maggi permissible lead content is 5

NULL Hypothesis H0 = lead = 5

Alternate hypothesis , H1 = lead < 5 or > 5

= symbol for Null Hypothesis , > or < for alternate hypothesis

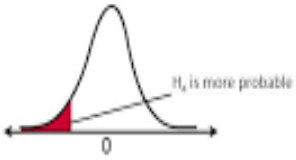

$\neq$ in $H_1$ → Two-tailed test → Rejection region on **both sides** of distribution

$<$ in $H_1$ → Lower-tailed test → Rejection region on **left side** of distribution

$>$ in $H_1$ → Upper-tailed test → Rejection region on **right side** of distribution

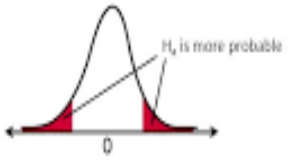

Right-tail test

$H_a$: $\mu >$ value



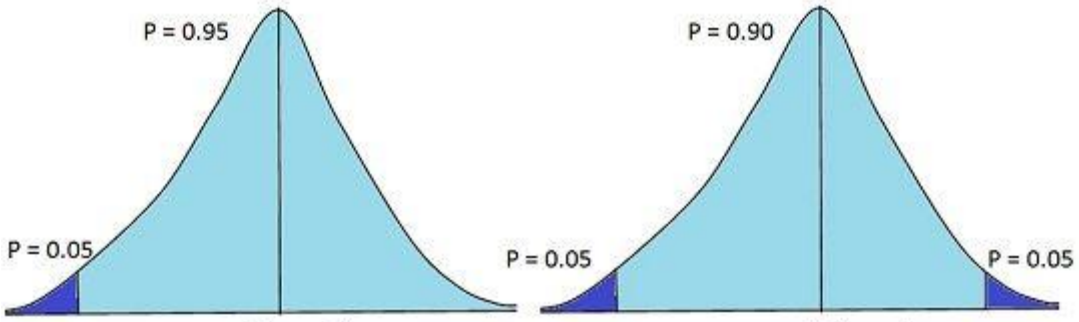

Left-tail test

$H_a$: $\mu <$ value

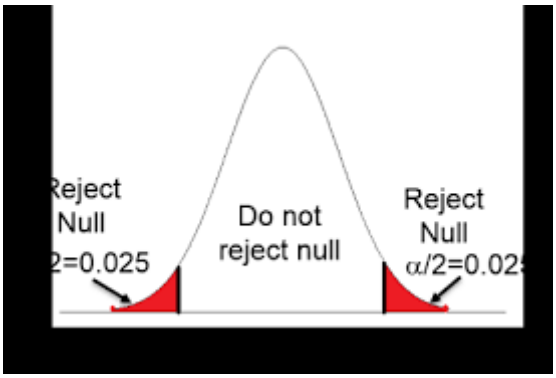


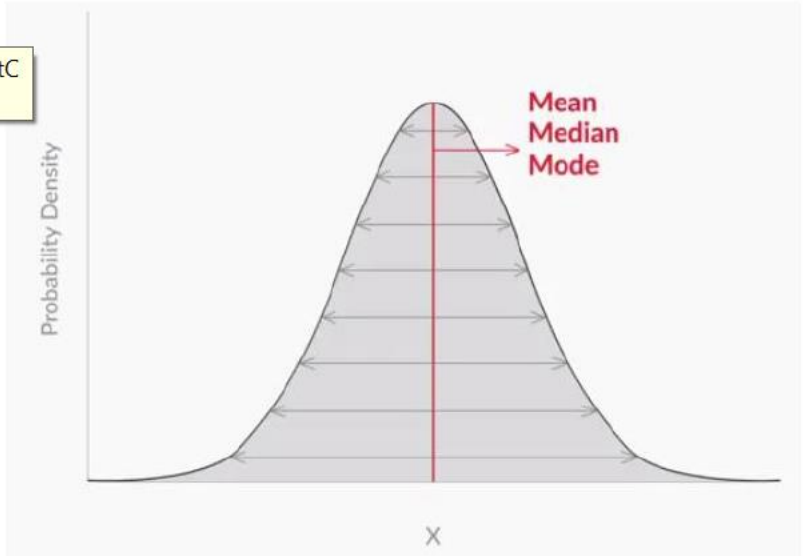

Two-tail test

$H_a$: $\mu \neq$ value




# One-tailed Test Vs Two-tailed Test

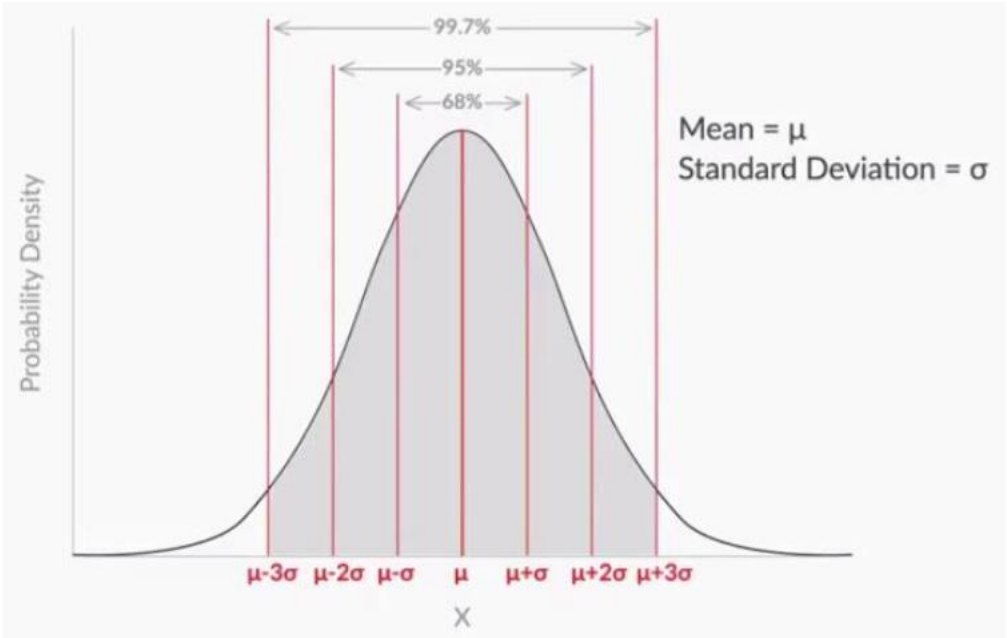





68.2% of the total data points lie in the range (Mean – Standard Deviation to Mean + Standard Deviation).

95.5% of the total data points lie in the range (Mean – 2*Standard Deviation to Mean + 2*Standard Deviation)

99.7% of the total data points lie in the range (Mean – 3*Standard Deviation to Mean + 3*Standard Deviation)



| Population/Sample | Term | Notation | Formula |
|---|---|---|---|
| Population<br>$(X_1, X_2, X_3, \ldots\ldots, X_N)$ | Population Size | N | Number of items/elements in the population |
| | Population Mean | $\mu$ | $\frac{\sum_{i=1}^{i=N} X_i}{N}$ |
| | Population Variance | $\sigma^2$ | $\frac{\sum_{i=1}^{i=N}(X_i - \mu)^2}{N}$ |
| Sample<br>$(X_1, X_2, X_3, \ldots\ldots, X_n)$<br>(Sample of Population) | Sample Size | n | Number of items/elements in the sample |
| | Sample Mean | $\bar{X}$ | $\frac{\sum_{i=1}^{i=n} X_i}{n}$ |
| | Sample Variance | $S^2$ | $\frac{\sum_{i=1}^{i=n}(X_i - \bar{X})^2}{n-1}$ |

| Population/Sample | Term | Notation | Formula |
|---|---|---|---|
| Population<br>$(X_1, X_2, X_3, \ldots\ldots, X_N)$ | Population Size | N | Number of items/elements in the population |
| | Population Mean | $\mu$ | $\frac{\sum_{i=1}^{i=N} X_i}{N}$ |
| | Population Variance | $\sigma^2$ | $\frac{\sum_{i=1}^{i=N}(X_i - \mu)^2}{N}$ |
| Sample<br>$(X_1, X_2, X_3, \ldots\ldots, X_n)$<br>(Sample of Population) | Sample Size | n | Number of items/elements in the sample |
| | Sample Mean | $\bar{X}$ | $\frac{\sum_{i=1}^{i=n} X_i}{n}$ |
| | Sample Variance | $S^2$ | $\frac{\sum_{i=1}^{i=n}(X_i - \bar{X})^2}{n-1}$ |
| Sampling Distribution of the Sample Mean<br>$(\bar{X}_1, \bar{X}_2, \bar{X}_3, \ldots\ldots, \bar{X}_k)$<br>(k Sample Means) | Sampling Distribution's Size | No convention (We have used k, but that is not a norm) | |
| | Sampling Distribution's Mean<br>(mean of sample means) | $\mu_{\bar{X}}$ | $\mu_{\bar{X}} = \mu$ |
| | Sampling Distribution's Standard Deviation | S.E. (Standard Error) | S.E. = $\sigma/\sqrt{n}$ |

So, there are two important properties for a sampling distribution of the mean:

1. **Sampling distribution's mean** ($\mu_{\bar{X}}$) = **Population mean** ($\mu$)
2. Sampling distribution's standard deviation (**Standard error**) = $\sigma/\sqrt{n}$, where $\sigma$ is the population's standard deviation and n is the sample size

**Z-Score**

$$Z = \frac{X - \mu}{\sigma}$$

**X – pop mean , u – sample mean , sigma/root(n) , sigma : pop std deviation , n – sample number**

**Pvalue – 1 – Z.**

**Confidence Interval:**

$$(\bar{X} - \frac{Z^*S}{\sqrt{n}}, \bar{X} + \frac{Z^*S}{\sqrt{n}})$$

| Confidence Level | Z* |
|---|---|
| 90% | ±1.65 |
| 95% | ±1.96 |
| 99% | ±2.58 |

X - Total Mean

Z- z score

S – overall std deviation

N – total number of sample

# Linear Regression

Two types of linear regression:

- Simple linear regression
- Multiple linear regression

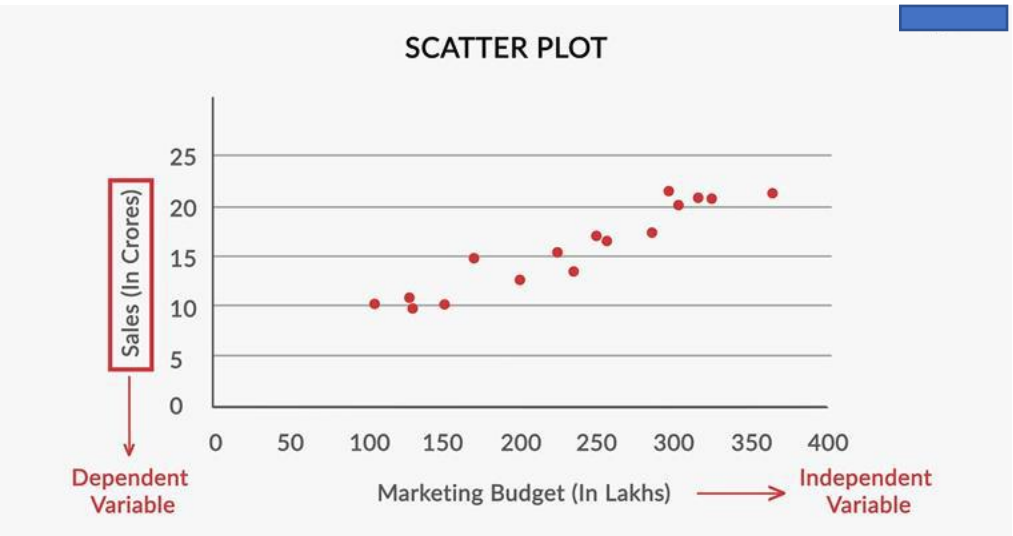
## Simple Linear regression:

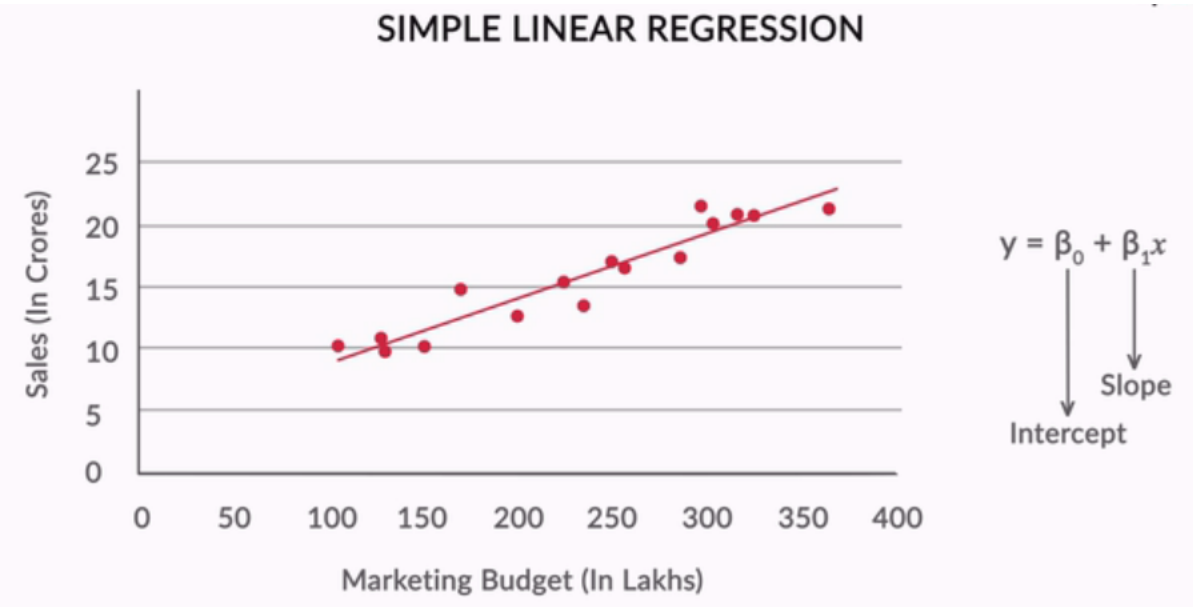
Equation of a straight line:

slope

$$y = mx + b$$

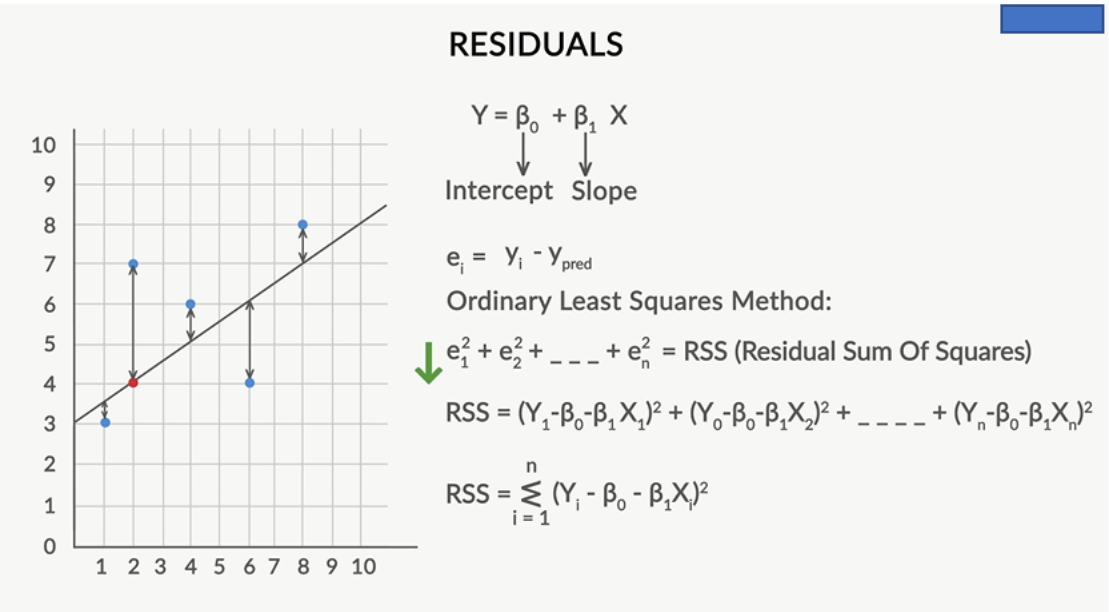
y-intercept






## Best Fit Line:

The best-fit line is found by minimising the expression of RSS (Residual Sum of Squares) which is equal to the sum of squares of the residual for each data point in the plot. Residuals for any data point is found by subtracting predicted value of dependent variable from actual value of dependent variable:



## Excel Demonstration:

slr_excel_demonstration.xlsx

## R2 or Coefficient of Determination

**R2 Formula**

- $R2 = 1 - \frac{RSS}{TSS}$

Where

RSS= Residual sum of square

TSS= Sum of errors of the data from mean

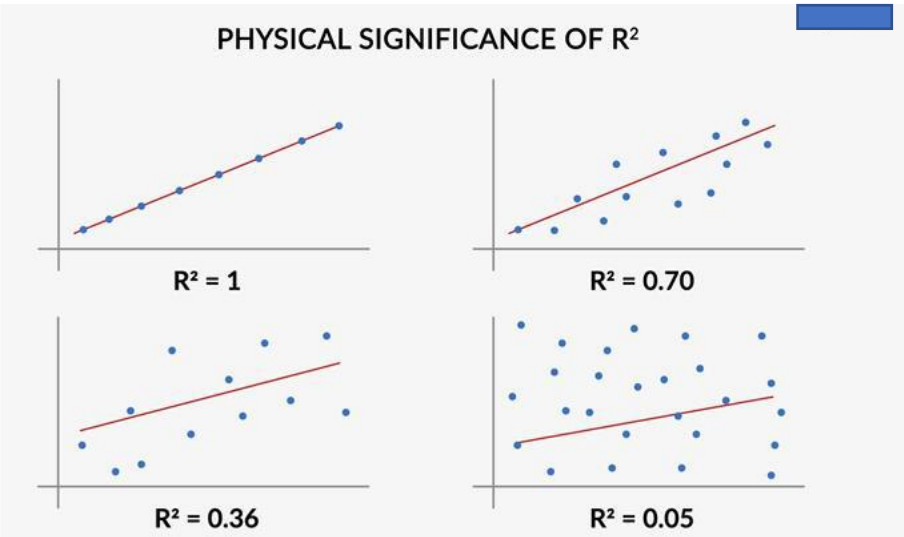
$$RSS = \sum_{i=1}^{n} (y_i - (\alpha + \beta x_i))^2$$

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where

$R^2$ Sample R-Squared
$N$ Total Sample Size
$p$ Number of independent variable



## Simple linear regression model in Python:

1. Import the data set and required libraries.

```
import pandas as pd
advertising = pd.read_csv("tvmarketing.csv")
```

2. Understanding the data frame.

```
# To display the first 5 rows
advertising.head()
# To display the last 5 rows
advertising.tail()
3. Preparing X and y
```

3. Splitting data into train and test

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7 , random_state=100)
```

4. Performing Linear Regression

```
# import LinearRegression from sklearn
from sklearn.linear_model import LinearRegression
# Representing LinearRegression as lr(Creating LinearRegression Object)
lr = LinearRegression()
#You don't need to specify an object to save the result because 'lr' will take the results of the fitted model.
lr.fit(X_train, y_train)
```

5. Coefficients calculation

```
# Print the intercept and coefficients
print(lr.intercept_)
print(lr.coef_)
```

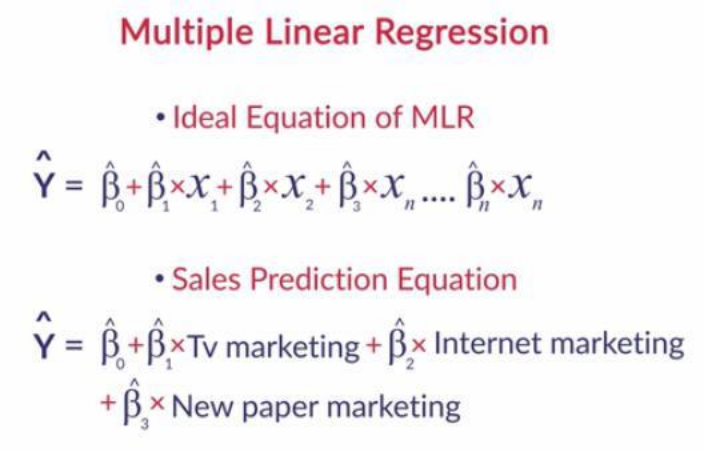
.

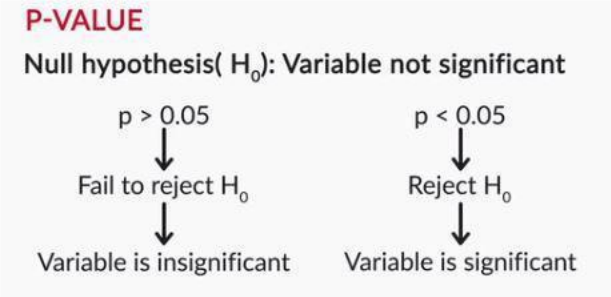
6.Making predictions

```
# Making predictions on the testing set
y_pred = lr.predict(X_test)
```

## Multiple Linear Regression:

Multiple linear regression is a statistical technique to understand the relationship between one dependent variable and several independent variables (explanatory variables).




**Variance inflation factor** (VIF).

$$VIF_i = \frac{1}{1 - R_i^2}$$

**Regression Analysis: HeatFlux versus East, South, North**

Coefficients

| Term | Coef | SE Coef | T-Value | P-Value | VIF |
|---|---|---|---|---|---|
| Constant | 389.2 | 66.1 | 5.89 | 0.000 | |
| East | 2.12 | 1.21 | 1.75 | 0.092 | 1.12 |
| South | 5.318 | 0.963 | 5.52 | 0.000 | 1.21 |
| North | -24.13 | 1.87 | -12.92 | 0.000 | 1.09 |

Regression Equation

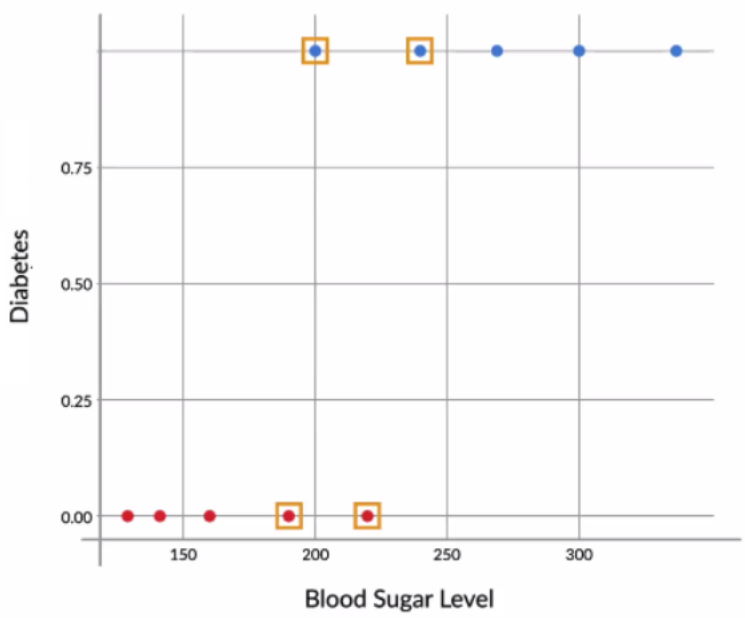
HeatFlux = 389.2 + 2.12 East + 5.318 South - 24.13 North

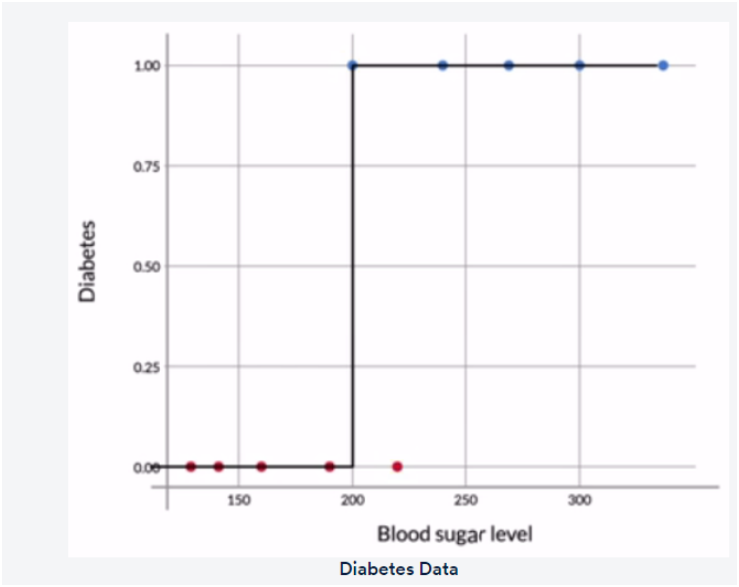
Remove variables with high VIF (>2 generally) and which are insignificant (p>0.05), one by one

# Logistic Regression

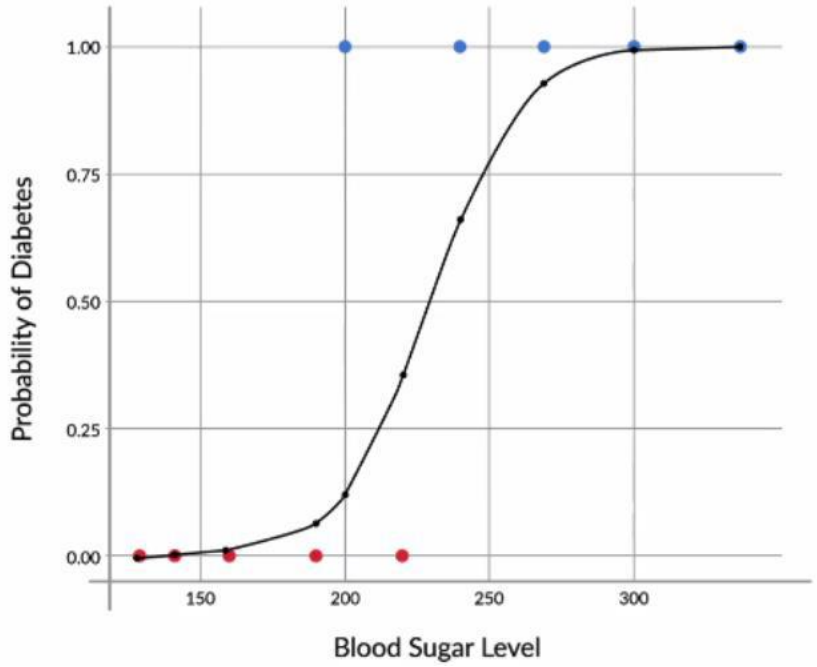
Logistic Regression is also called as binary classification

- Binary classification
- Sigmoid function
- Likelihood function
- Odds and log odds

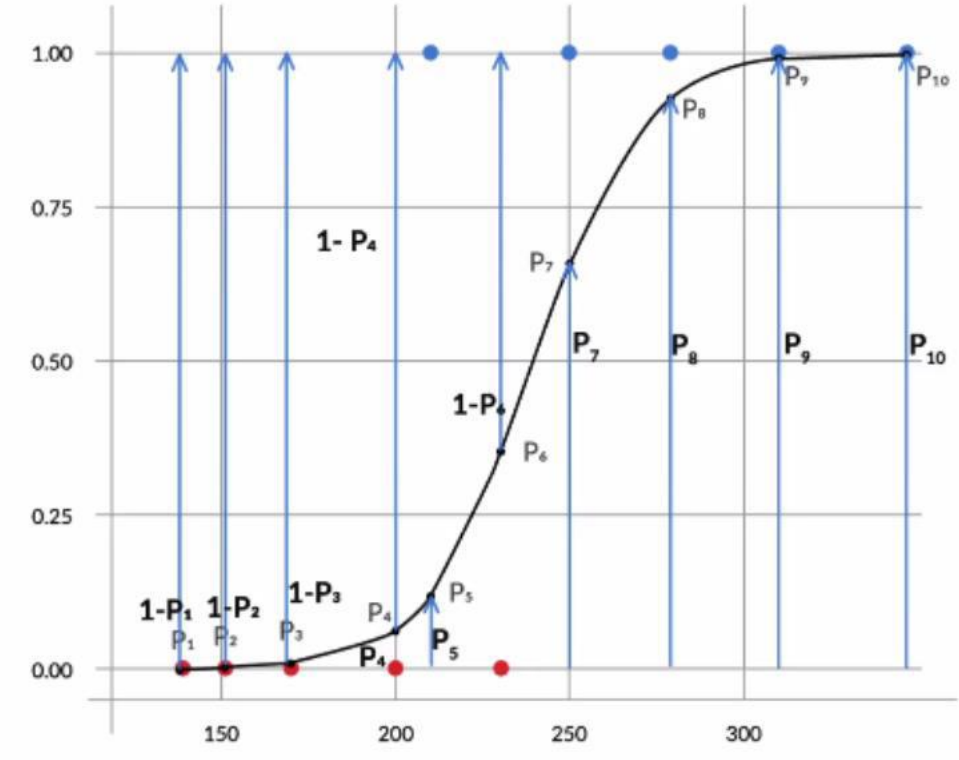



Diabetes Data




$$P(\text{Diabetes}) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

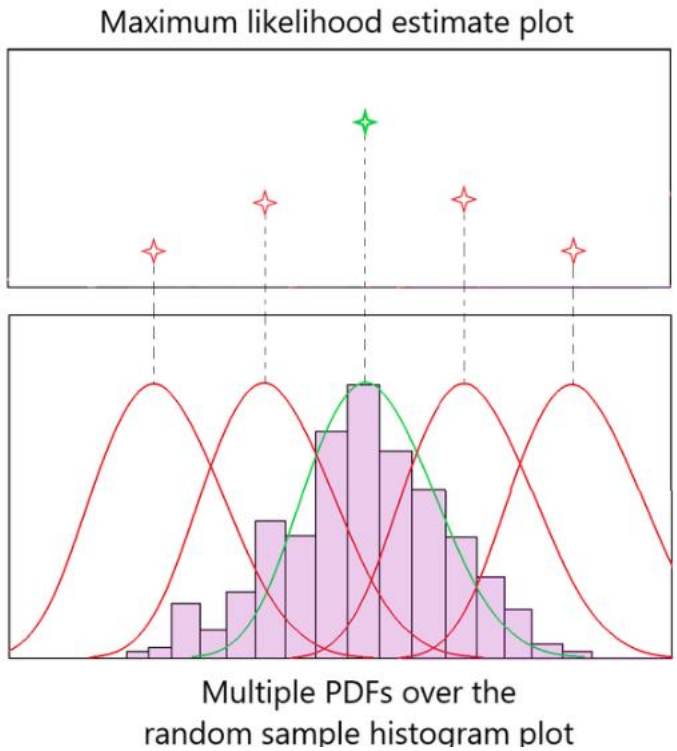


Likelihood = $(1-P_1)(1-P_2)(1-P_3)(1-P_4)(P_5)(1-P_6)(P_7)(P_8)(P_9)(P_{10})$

$$P = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\cdots)}}$$

**Logistic Regression - Optimisation Methods (Optional)**

The question is - how do you find the optimal values of β0 and β1 such that the likelihood function is maximized? The optimisation methods used to do that are an optional part of the course (maximum likelihood estimation, or MLE).




# Odds and Log Odds

**Probability** is the probability an event happens. For example, there might be an 80% chance of rain today.
**Odds** (more technically the *odds of success*) is defined as probability of success/probability of failure. So the odds of a success (80% chance of rain) has an accompanying odds of failure (20% chance it doesn't rain); as an equation (the "**odds ratio**"), that's .8/.2 = 4.
**Log odds** is the logarithm of the odds. Ln(4) = 1.38629436 ≅ 1.386.

Odds :

$$P=1 / 1+e^{-(\beta 0+\beta 1x)}$$

$$1 - P= e^{-(\beta 0+\beta 1x)} / 1+e^{-(\beta 0+\beta 1x)}$$

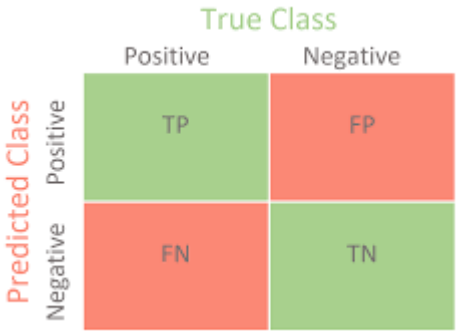
$$p/1-p = e^{(\beta 0+\beta 1x)}$$

log odds:

$$\ln(P1-P)=\beta_0+\beta_1 x$$

< excel on odds log odds >

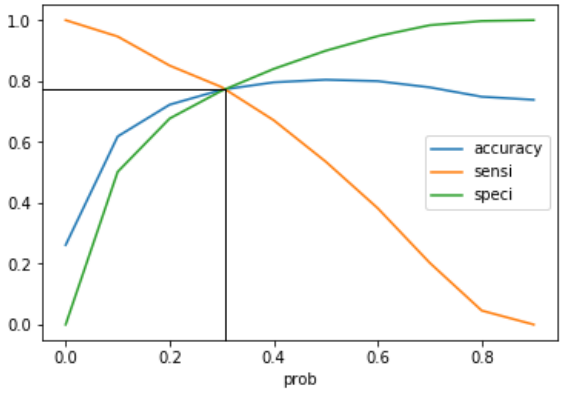
Confusion Matrix:



$$Sensitivity = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{TN+FP}$$

$$Accuracy = \frac{Correctly\ Predicted\ Labels}{Total\ Number\ of\ Labels}$$




**Key Metrics:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

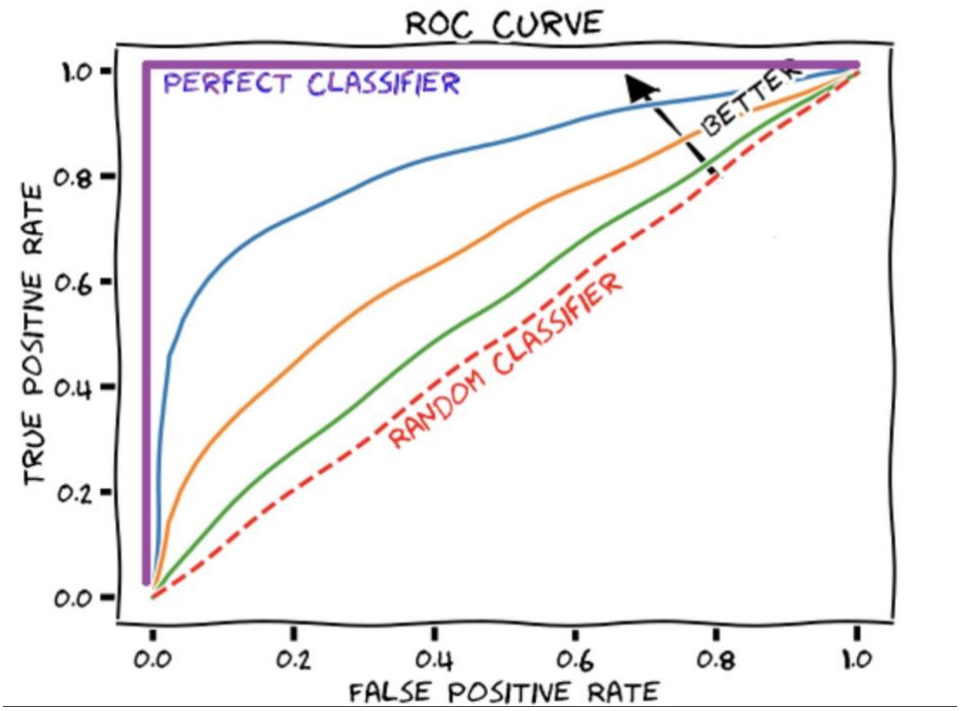
# ROC curve

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

$$\text{TPR (sensitivity)} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN}+\text{FP}}$$




# Naïve Baise

## Conditional probability:

$$\boldsymbol{P(A|B) = P(A \cap B)/P(B)}$$

P (A|B) is the probability of event A occurring, given that event B occurs. Example: given that you drew a red card, what's the probability that it's a four (p(four|red))=2/26=1/13. So out of the 26 red cards (given ared card), there are two fours so 2/26=1/13.

# Joint probability:

$$P(A \cap B) = P(A|B) * P(B)$$

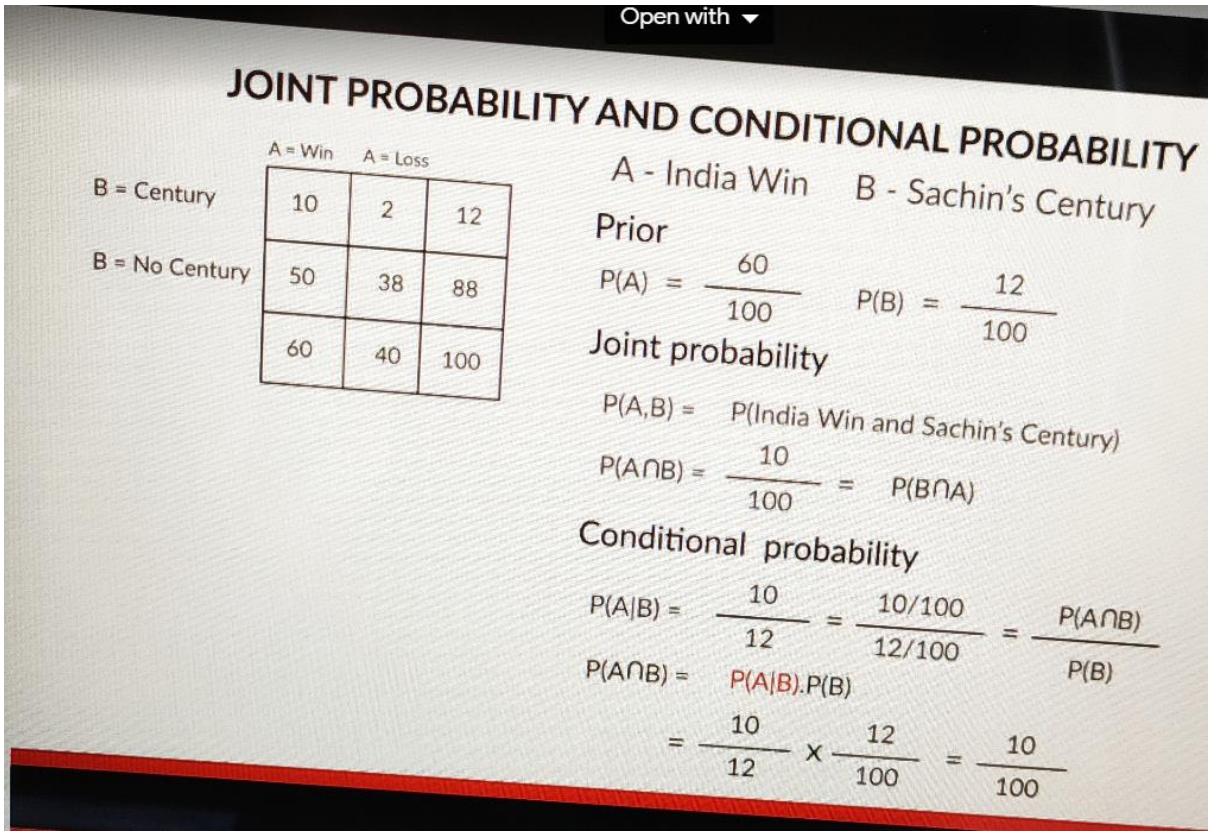
P (A and B). The probability of event A **and** event B occurring. It is the probability of the intersection of twoor more events. The probability of the intersection of A and B may be written p(A ∩ B). Example: the probability that a card is a four and red =p(four and red) = 2/52=1/26. (There are two red fours in a deck of52, the 4 of hearts and the 4 of diamonds).

**Bayes Theorem :** Given Probability of an event B occurring given event A has already occurred and individual Probabilities of A and B we can find the reverse conditional probability P(A|B) by using what is called BayesTheorem which is shown below.

$$P(A|B) = \frac{P(B|A) \ . P(A)}{P(B)}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Bayes Theorem



Test Dataset

| | Document | Class |
|---|---|---|
| 0 | UpGrad is a great educational institution. | education |
| 1 | Educational greatness depends on ethics | education |
| 2 | A story of great ethics and educational greatness | education |
| 3 | Sholey is a great cinema | cinema |
| 4 | good movie depends on good story | cinema |

# DICTIONARY/VOCABULARY

Dictionary before Stop Word removal

| |
|---|
| 0 : and |
| 1 : cinema |
| 2 : depends |
| 3 : educational |
| 4 : ethics |
| 5 : good |
| 6 : great |
| 7 : greatness |
| 8 : institiution |
| 9 : is |
| 10 : movie |
| 11 : of |
| 12 : on |
| 13 : sholey |
| 14 : story |
| 15 : upgrad |

Stop Words

| |
|---|
| 0 : and |
| 9 : is |
| 11 : of |
| 12 : on |

Dictionary after Stop Word removal

| |
|---|
| 0 : cinema |
| 1 : depends |
| 2 : educational |
| 3 : ethics |
| 4 : good |
| 5 : great |
| 6 : greatness |
| 7 : institiution |
| 8 : movie |
| 9 : sholey |
| 10 : story |
| 11 : upgrad |

# BAG OF WORDS REPRESENTATION

Dictionary/Vocabulary

Documents

| | cinema | depends | educational | ethics | good | great | greatness | insitution | movie | sholey | story | upgrad |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

5th Sentence: good movie depends on good story

good movie story depends good on

[cinema, . . . . . ., upgrad]

$$D^{education} = \begin{bmatrix} 0,0,1,0,0,1,0,1,0,0,0,1 \\ 0,1,1,1,0,0,1,0,0,0,0,0 \\ 0,0,1,1,0,1,1,0,0,0,1,0 \end{bmatrix} \quad 13$$

$$0,1,3,2,0,2,2,1,0,0,1,1$$

$$D^{cinema} = \begin{bmatrix} 1,0,0,0,0,1,0,0,0,1,0,0 \\ 0,1,0,0,2,0,0,0,1,0,1,0 \end{bmatrix} \quad 8$$

$$1,1,0,0,2,1,0,0,1,1,1,0$$

**Prior**

$P(education) = 3/5$

$P(cinema) = 2/5$

$P(education|w_1,w_2,.....,w_n)$
$P(cinema|w_1,w_2,.....,w_n)$ ] **Posterior**

$$P(cinema|w_1,w_2,.....,w_n) = \frac{P(w_1,w_2,.....,w_n|class)\ P(class)}{P(w_1,w_2,.....,w_n)}$$

$$= P(w_1|c)\ P(w_2|c).....P(w_n|c) \times P(c)$$

| | $n_{education}^{(w)}$ | p(w\|c = education) | $n_{cinema}^{(w)}$ | p(w\|c = cinema) |
|---|---|---|---|---|
| Dictionary/Vocabulary | | | | |
| $w_1$ = cinema | 0 | 0 | 1 | 1/8 |
| $w_2$ = depends | 1 | 1/13 | 1 | 1/8 |
| $w_3$ = educational | 3 | 3/13 | 0 | 0 |
| $w_4$ = ethics | 2 | 2/13 | 0 | 0 |
| $w_5$ = good | 0 | 0 | 2 | 2/8 |
| $w_6$ = great | 2 | 2/13 | 1 | 1/8 |
| $w_7$ = greatness | 2 | 2/13 | 0 | 0 |
| $w_8$ = institiution | 1 | 1/13 | 0 | 0 |
| $w_9$ = movie | 0 | 0 | 1 | 1/8 |
| $w_{10}$ = sholey | 0 | 0 | 1 | 1/8 |
| $w_{11}$ = story | 1 | 1/13 | 1 | 1/8 |
| $w_{12}$ = upgrad | 1 | 1/13 | 0 | 0 |

$P(education|w_1,w_2,.....,w_n)$

$P(cinema|w_1,w_2,.....,w_n) \propto P(w_1,w_2,.....,w_n|c)\ P(cinema)$

$\propto P(w_1|c) \times P(w_2|c).....\times P(cinema)$

$P(w_3|cinema) = 0$

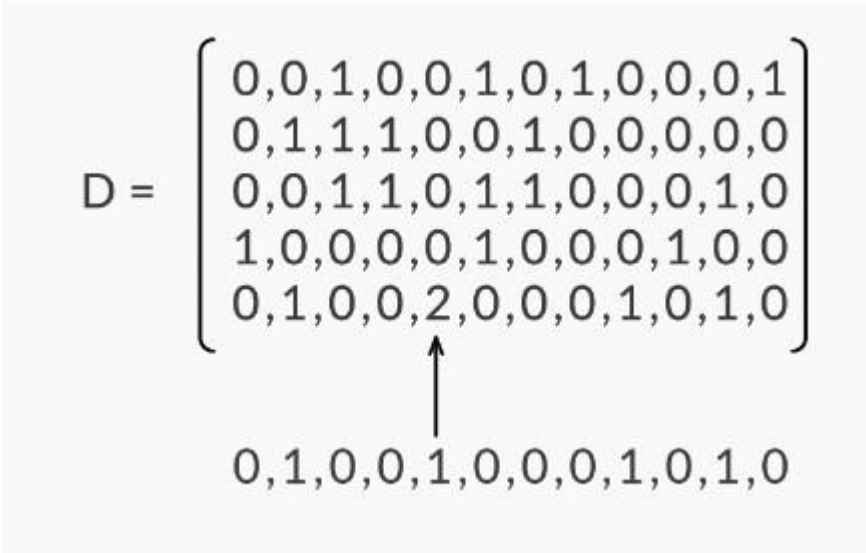
$P(w_3|education) = 3/13$

Great story – Education = 2/13 * 1/13 * 3/5 = 0.007

Great Story – cinema = 1/8 * 1/8 * 2/5 = 0.006

# Introduction to Bernoulli Theorem

- Recall the difference between Multinomial and Bernoulli way of building feature vector. UnlikeMultinomial way which is concerned about the no. of occurrences of the word in the class , in Bernoulli we are just concerned about whether the word is present or not.

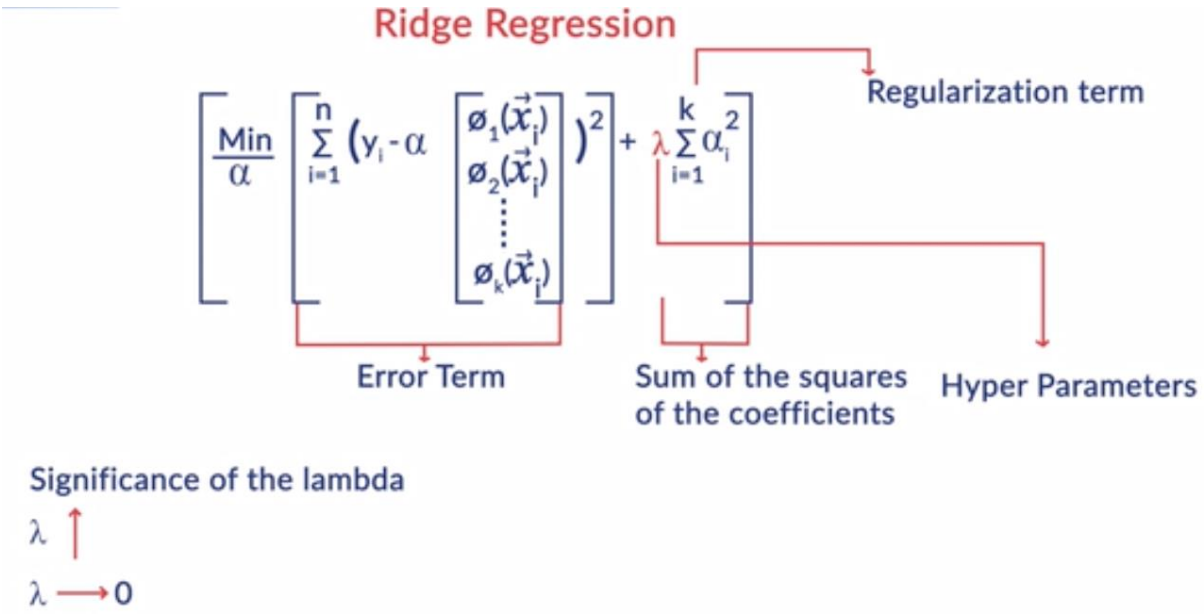



# Regularized Regression

A predictive model has to be as simple as possible, but no simpler. There is an important relationship
between the complexity of a model and its usefulness in a learning context because of the following
reasons:
• Simpler models are usually more generic and are more widely applicable (are generalizable)
• Simpler models require fewer training samples for effective training than the more complex ones
**Regularization** is a process used to create an optimally complex model, i.e. a model which is as simple as
possible while performing well on the training data.
Through regularization, the algorithm designer tries to strike the delicate balance between keeping
the model simple, yet not making it too naive to be of any use.
The regression does not account for model complexity - it only tries to minimize the error (e.g. MSE),
although if it may result in arbitrarily complex coefficients. On the other hand, in regularized regression,
the objective function has two parts - the **error term** and the **regularization term**.

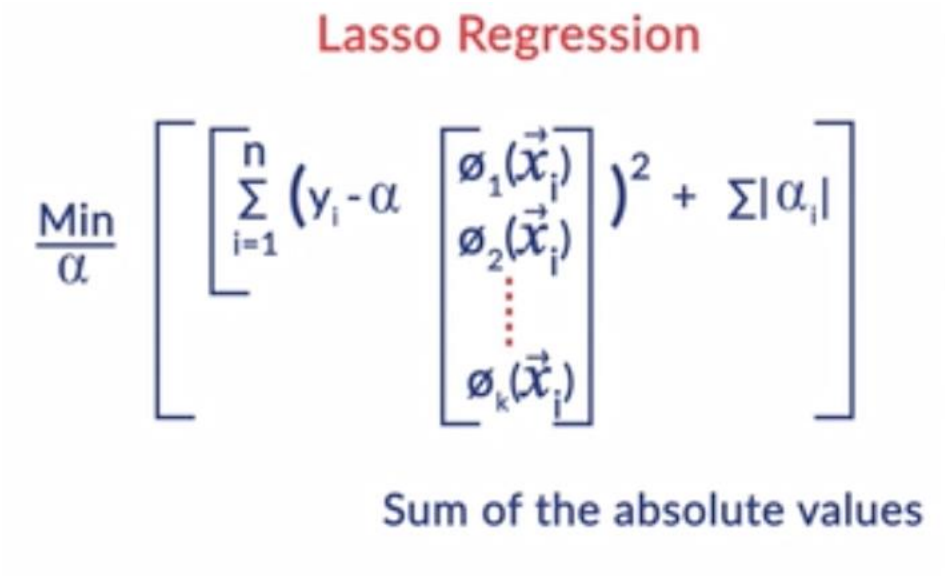
**Ridge Regression:**

In ridge regression, an additional term of "sum of the squares of the coefficients" is added to the cost
function along with the error term




$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

**Lasso Regression:**

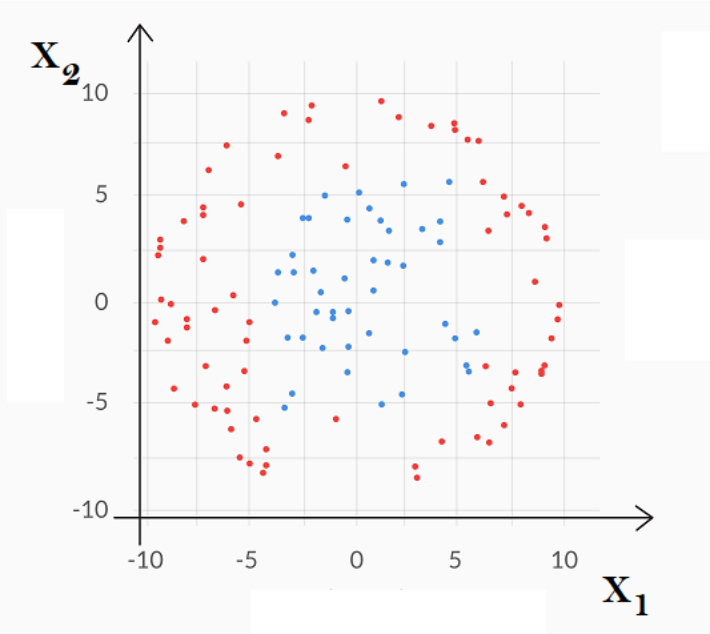



$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}$$

# SVM (Support vector Machines)

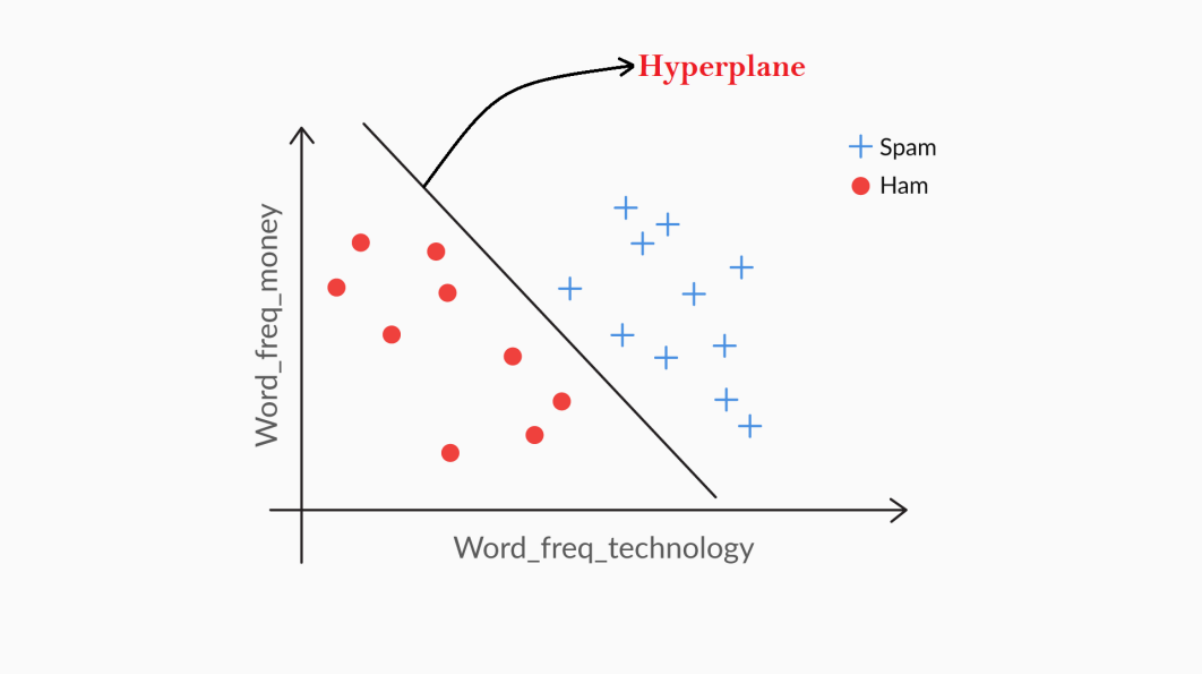
Support Vector Machine (SVM) is an advanced machine learning technique which has a unique way of solving complex problems such as image recognition, face detection, voice detection etc. As you will learn in this session, SVMs solves the Pioi;; 98oblem of nonlinearity through kernels.

For instance, if you have a data as shown in the figure below, SVMs can handle it easily and that's how SVM distinguishes from logistic regression.



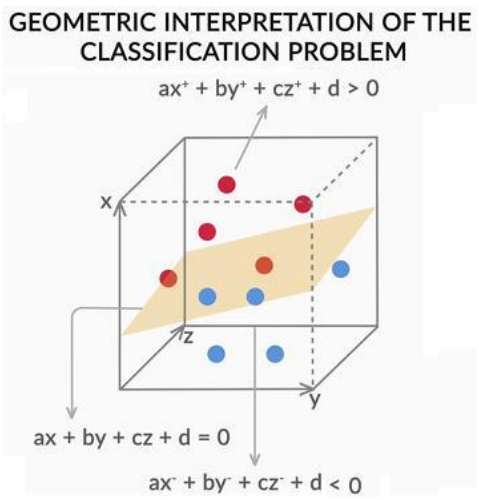

## Concept of Hyperplane in 2D

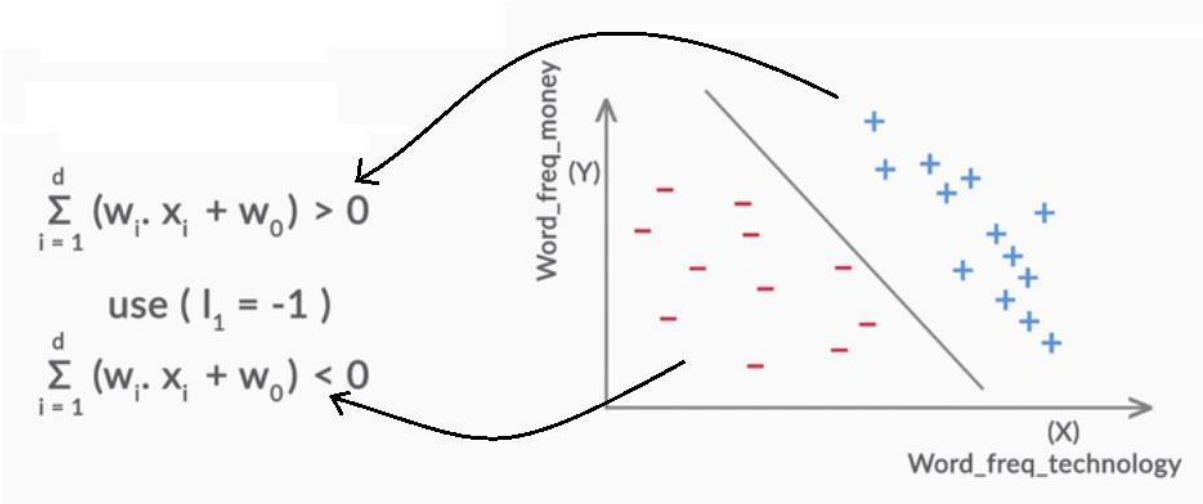
Before you move on to support vector machines, you need to understand the concept of hyperplanes. Essentially, it is a boundary which classifies the data set (classifies Spam email from the ham ones). It could be lines, 2D planes, or even n-dimensional planes that are beyond our imagination.
A line that is used to classify one class from another is called a hyperplane. In fact, it is the model you're trying to build as shown in the figure below:



## Concept of Hyperplane in 3D

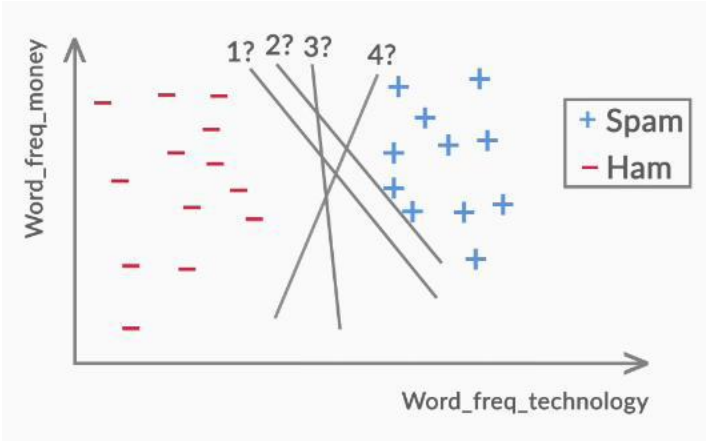
In 3 dimensions (refer to figure below), the hyperplane (light orange) will be a plane with an expression of **ax+by+cz+d = 0**. The plane divides the data set into two halves. Data points above the plane represent one class (red), while data points below the plane represent the other class (blue).



## Maximal Margin Classifier

There could be multiple lines(Hyperplanes) possible which perfectly separate the two classes as shown in the figure below. But the best line, is the one which maintains the largest possible equal distance from the nearest points of both the classes so for the separator to be optimal, the margin or the distance of the nearest point to the separator should be maximum. This is called **Maximal Margin classifier**.




The mathematical formulation requires **two major constraints** that need to be taken into account while **maximising the margin**. They are

$(li*(Wi.Yi)) \geqslant M$

where,

li = label (1, -1)

Wi = coefficient of attributes

Yi = data points of all the attributes in each row

$$X + 2y + 3z = 0$$

$$2x + 4y + 6z = 0$$

$$2 * (X + 2y + 3z) = 0$$

- The standardisation of coefficients such that the summation of the square of the coefficients of all the attributes is equal to 1. For example, if you have 20 attributes, then the summation of square of the coefficients should be $\sum 20i=0(W2i)=1$

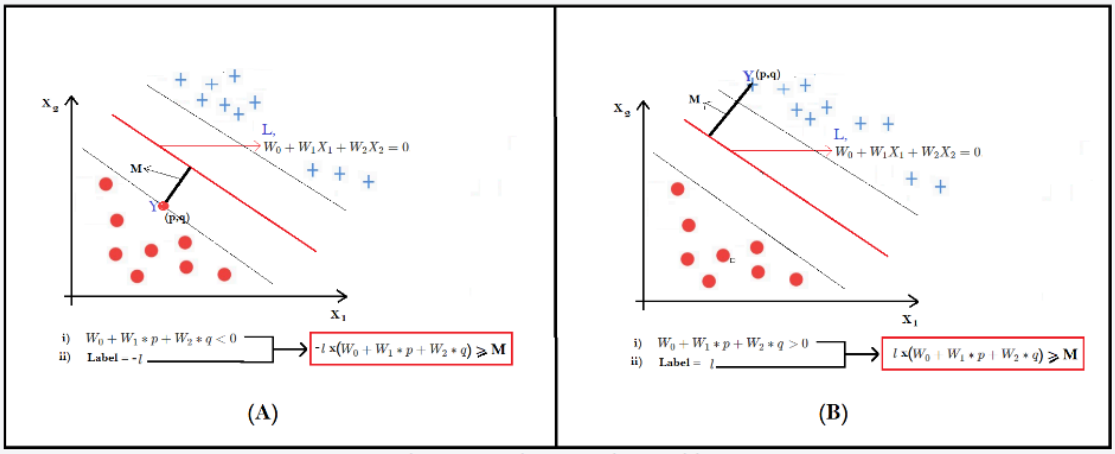
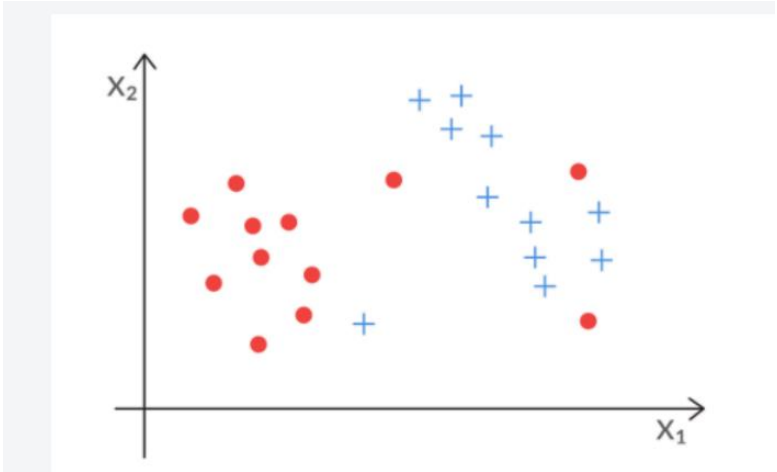


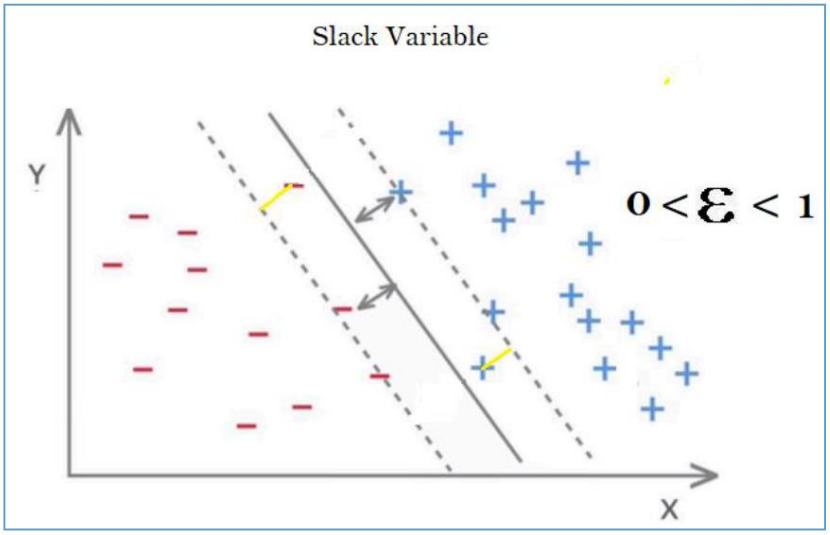
Figure-2:Maximal Margin Classifier

## The Soft Margin Classifier

The **Support Vector Classifier** essentially allows certain points to be deliberately misclassified. By doing this, it is able to classify most of the points correctly in the unseen data and is also more robust.

The Support Vector Classifier is also called the **Soft Margin Classifier** because instead of searching for the margin that exactly classifies each and every data point to the correct class, the Soft Margin Classifier allows some observations to fall on the wrong side. The points which are close to the hyperplane are only considered for constructing the hyperplane and those points are called **support vectors**.





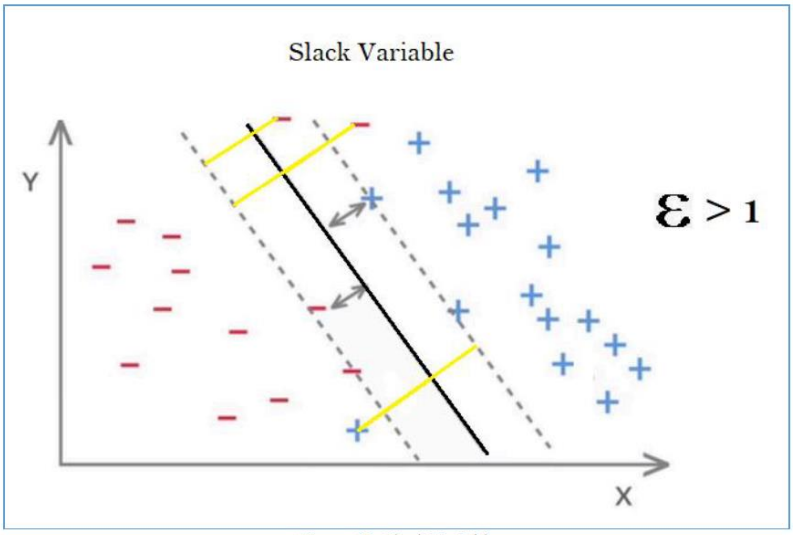
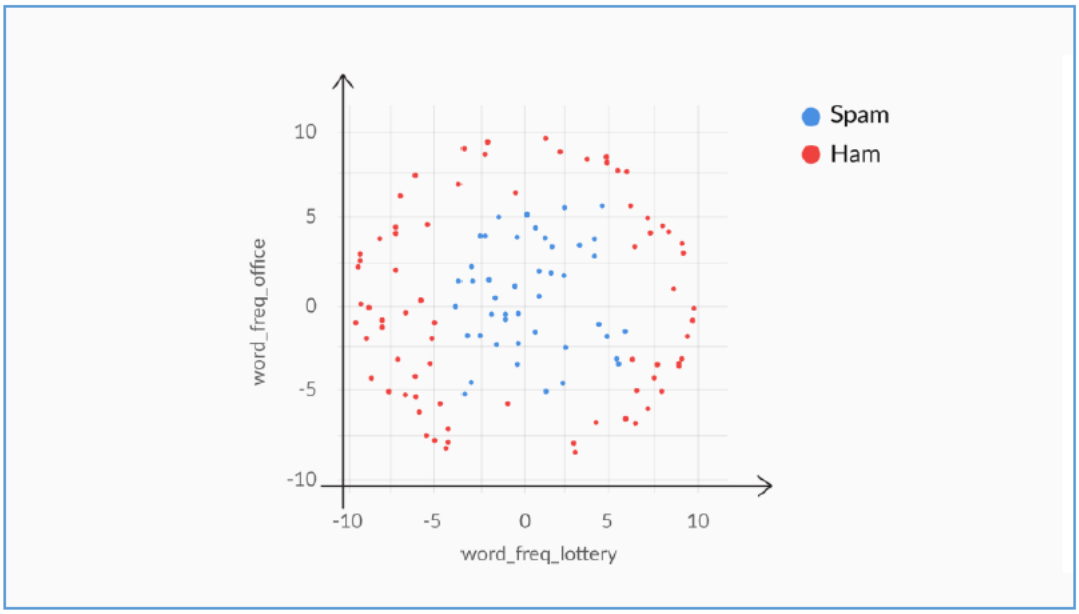




Figure 10: Slack Variable

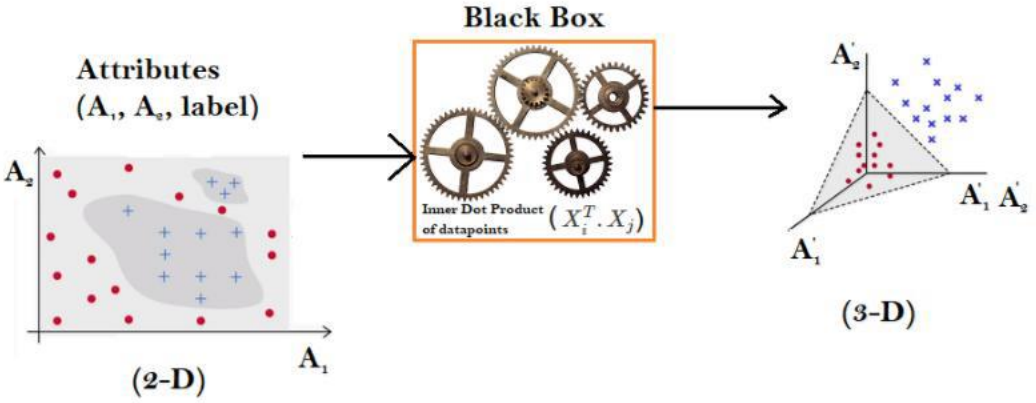
## Kernels

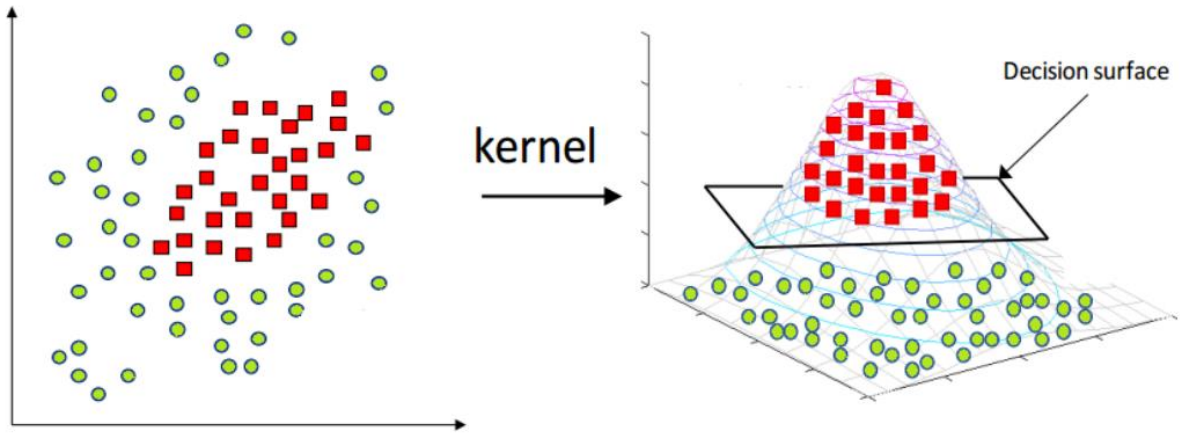
Kernels are one of the most interesting inventions in machine learning, partly because they were born through the creative imagination of mathematicians, and partly because of their utility in **dealing with non-linear datasets**

So far, we have learnt about hyperplanes, the Maximal Margin Classifier, and the Support Vector Classifier. All of these are linear models (since they use linear hyperplanes to separate the classes). However, many real-world data sets **are not separable by linear boundaries**. For instance, what if the distribution of data points looks like the figure given below?


Spam
Ham
word_freq_office
word_freq_lottery
10
5
0
-5
-10
-10
-5
0
5
10

## SVM -Kernels


Attributes
(A1, A2, label)
A2
A1
(2-D)
Black Box
Inner Dot Product
of datapoints
$(X_i^T . X_j)$
A'2
A'1
A'3
(3-D)

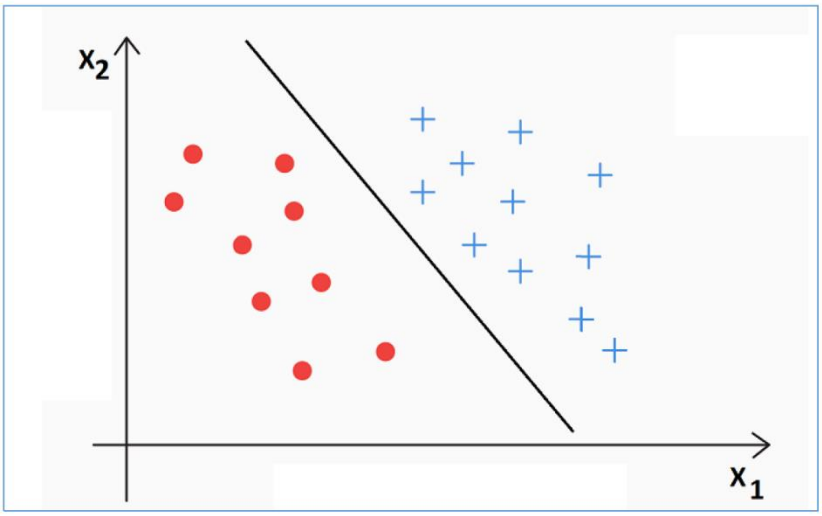


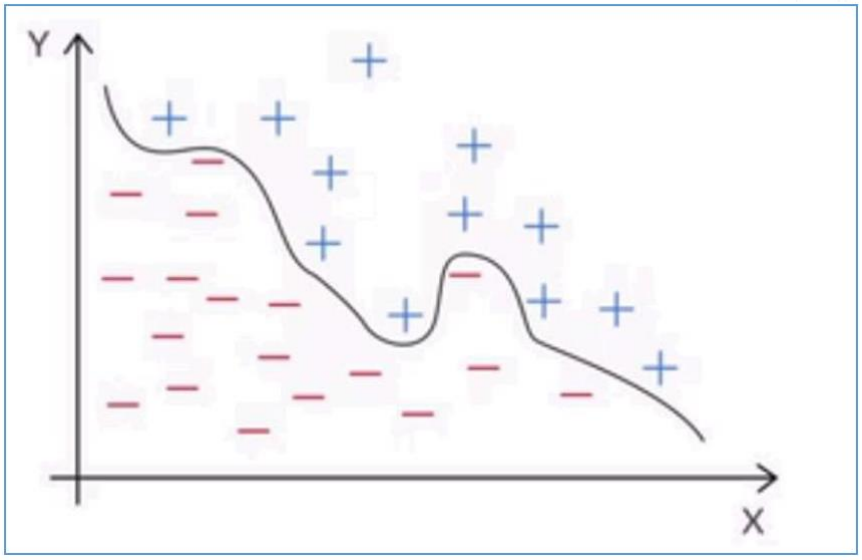

The **three most popular** types of kernel functions are:

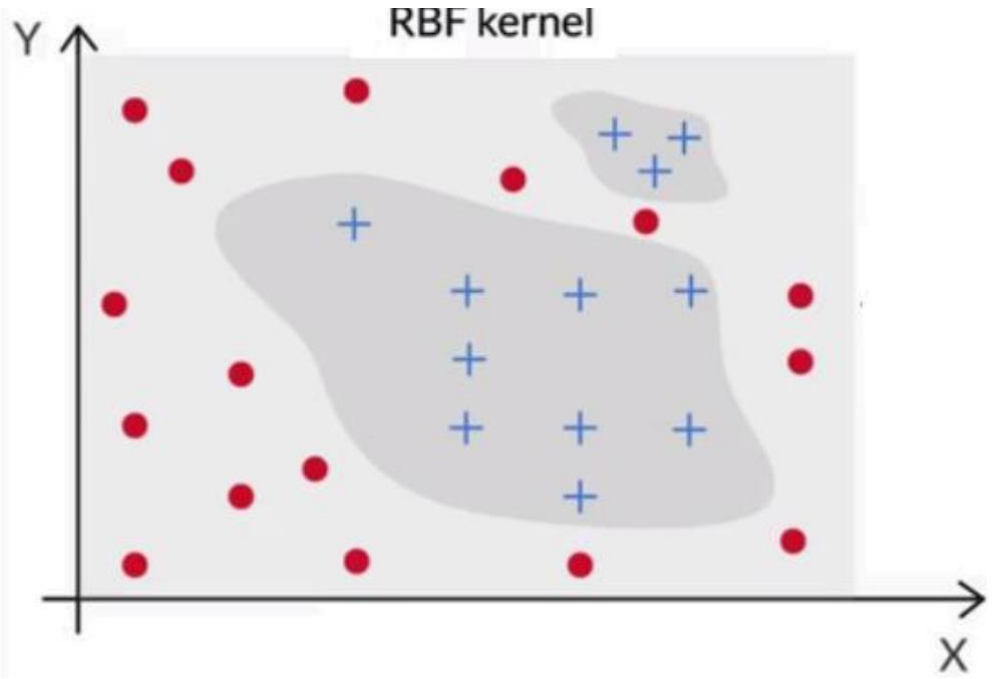
 **The linear kernel**: This is the same as the support vector classifier, or the hyperplane, without any transformation at all

 **The polynomial kernel**: It is capable of creating nonlinear, polynomial decision boundaries

 **The radial basis function (RBF) kernel**: This is the most complex one, which is capable of transforming highly nonlinear feature spaces to linear ones. It is even capable of creating elliptical (i.e. enclosed) decision boundaries

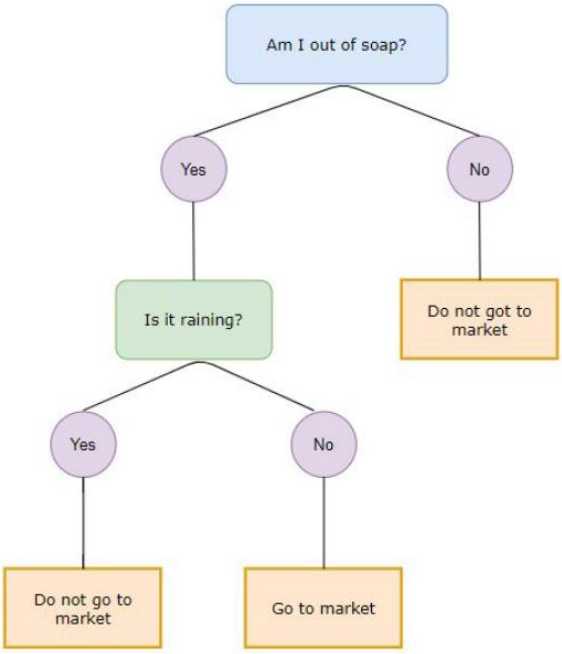


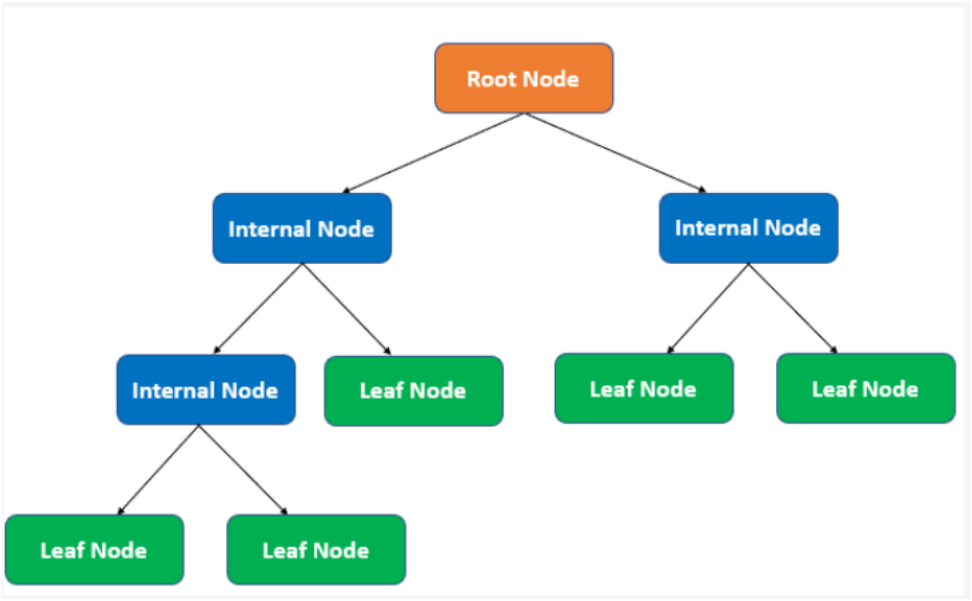

Y
X


RBF kernel
Y
X

# Tree Model

Decision Trees naturally represent the way we make decisions. Think of a machine learning model as a decision making engine that takes a decision on any given input object (data point). Imagine a doctor making a decision (the diagnosis) on whether a patient is suffering from a particular condition given the patient data, an insurance company making a decision on how much claims on a particular insurance policy needs to be paid out given the policy and the claim data, a company deciding on which role an applicant seeking a position in the company is eligible to apply for, based on the past track record and other details of the applicant, a real estate company aiming to optimise the selling price of the properties, based on important factors such as area, bedrooms, parking, etc. Solutions to each of these can be thought of as machine learning models trying to mimic the human decision making.

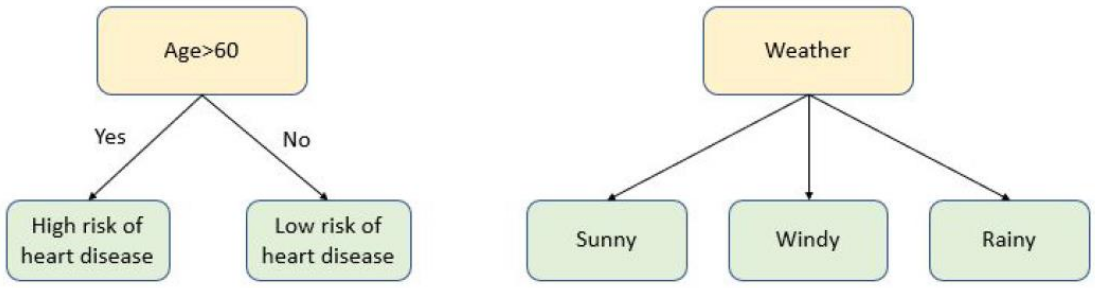





Componenets:

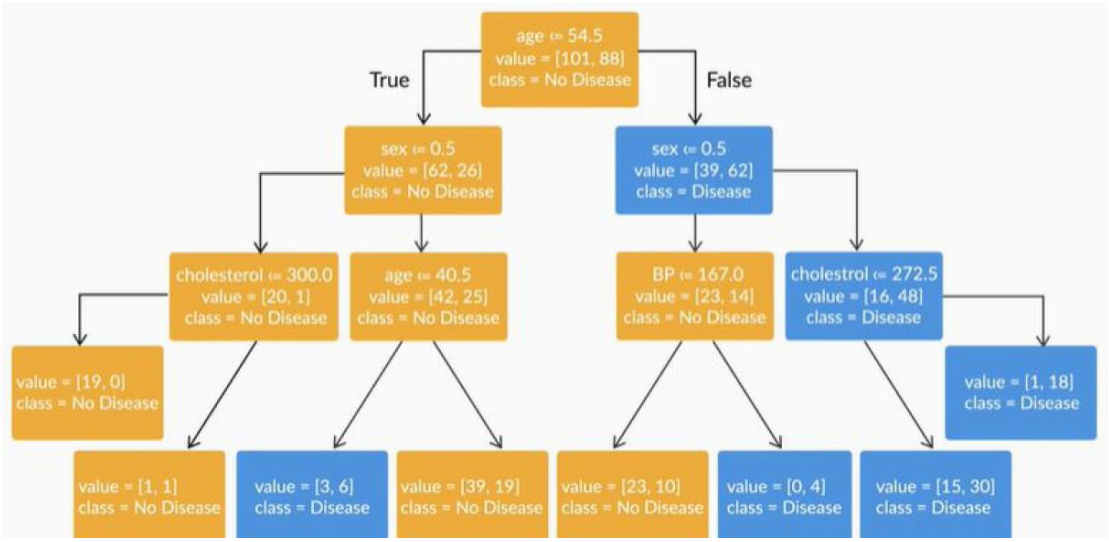
1. Binary decision tree: The test splits variables into exactly two partitions.

2. Multiway decision tree: The test splits variables into more than two partitions.

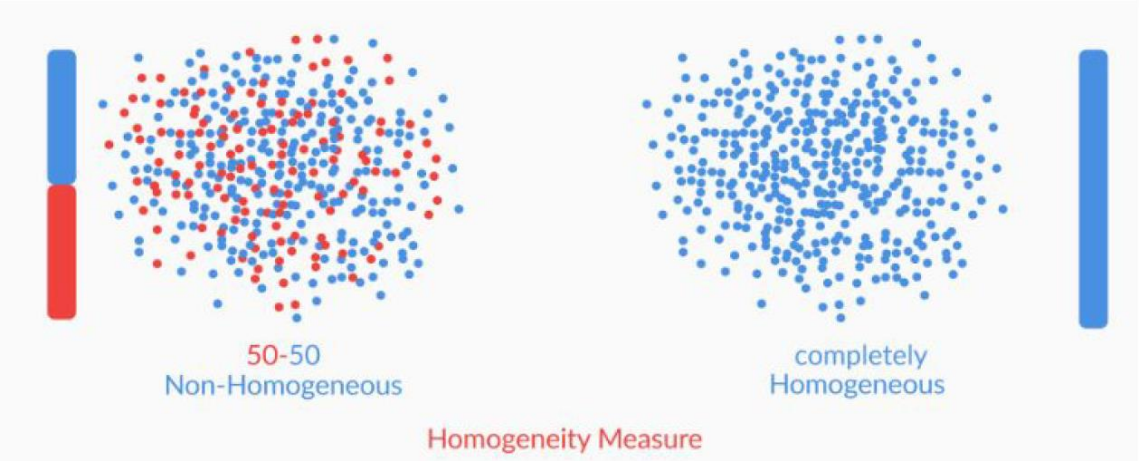


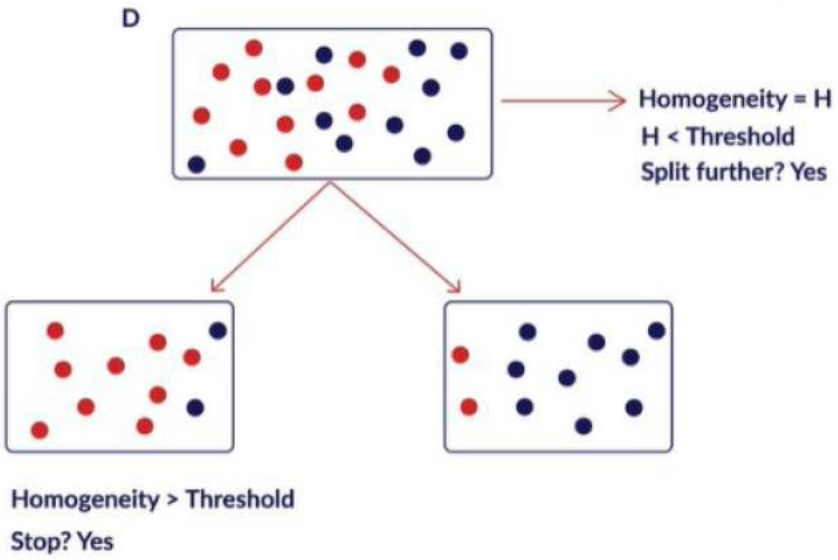

**Binary Decision Tree** **Multi-way Decision Tree**

Interpreting decision tree:


age <= 54.5
value = [101, 88]
class = No Disease
True
False
sex <= 0.5
value = [62, 26]
class = No Disease
sex <= 0.5
value = [39, 62]
class = Disease
cholesterol <= 300.0
value = [20, 1]
class = No Disease
age <= 40.5
value = [42, 25]
class = No Disease
BP <= 167.0
value = [23, 14]
class = No Disease
cholestrol <= 272.5
value = [16, 48]
class = Disease
value = [19, 0]
class = No Disease
value = [1, 18]
class = Disease
value = [1, 1]
class = No Disease
value = [3, 6]
class = Disease
value = [39, 19]
class = No Disease
value = [23, 10]
class = No Disease
value = [0, 4]
class = Disease
value = [15, 30]
class = Disease

Homogenity:


50-50
Non-Homogeneous
completely
Homogeneous
Homogeneity Measure




A tree can be split based on different rules of an attribute and these attributes can be categorical or continuous in nature. If an attribute is nominal categorical, then there are $2^{k-1}-1$ possible splits for this attribute, where k is the number of classes. In this case, each possible subset of categories is examined to determine the best split.

## Concept of splitting:

The classification error is calculated as follows:

- $E = 1 - max(p_i)$

The Gini index is calculated as follows:

- $G = \sum_{i=1}^{k} p_i(1-p_i)$

Entropy is calculated as follows:

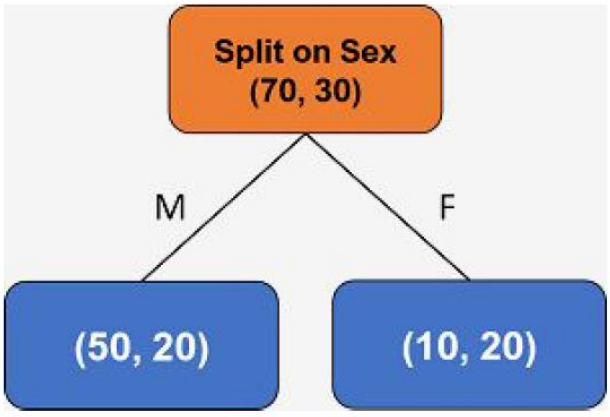
- $D = -\sum_{i=1}^{k} p_i . log_2(p_i)$,

where $p_i$ is the probability of finding a point with the label $i$, and $k$ is the number of classes.

Gini split:

| Features / Classes | Sex | | Total |
|---|---|---|---|
| | M | F | |
| No Disease | 50 | 10 | 60 |
| Disease | 20 | 20 | 40 |
| Total | 70 | 30 | 100 |

| Features / Classes | Cholesterol | | Total |
|---|---|---|---|
| | < 250 | > 250 | |
| No Disease | 50 | 10 | 60 |
| Disease | 10 | 30 | 40 |
| Total | 60 | 40 | 100 |




**[Note that (x, y) on any node means (# Label 0, # Label 1)]**

Now the probabilities of the two classes within the male subset comes out to be:

$p0=50/70=0.714$ and $p1=20/70=0.286$

Now using the same formula, Gini impurity for males becomes:

$$0.714(1-0.714)+0.286(1-0.286)=0.41$$

Let's now take the other case i.e. the child node containing females, where there are 30 females out of which 10 belong to class 0 having no heart disease and 20 belong to class 1 having a heart disease. The probabilities of the two classes within the female subset comes out to be:

$$p0=10/30=0.333 \quad \text{and} \quad p1=20/30=0.667$$

Now using the formula, Gini impurity for females becomes:

$$0.333(1-0.333)+0.667(1-0.667)=0.44$$

Now how do you get the overall impurity for the attribute 'sex' after the split? You can aggregate the Gini impurity of these two nodes by taking a weighted average of the impurities of the male and female nodes. So, you have -

$$pmale=70/100=0.7 \quad \text{and} \quad pfemale=30/100=0.3$$

This gives the Gini impurity after the split based on gender as:

$$0.7\times0.41+0.3\times0.44=0.42$$

No disease : 60 (Class 0)
Disease : 40 (Class 1)

Expressing this in terms of probabilities you get:

$$p_0 = \frac{60}{60+40} = 0.6 \qquad p_1 = \frac{40}{60+40} = 0.4$$

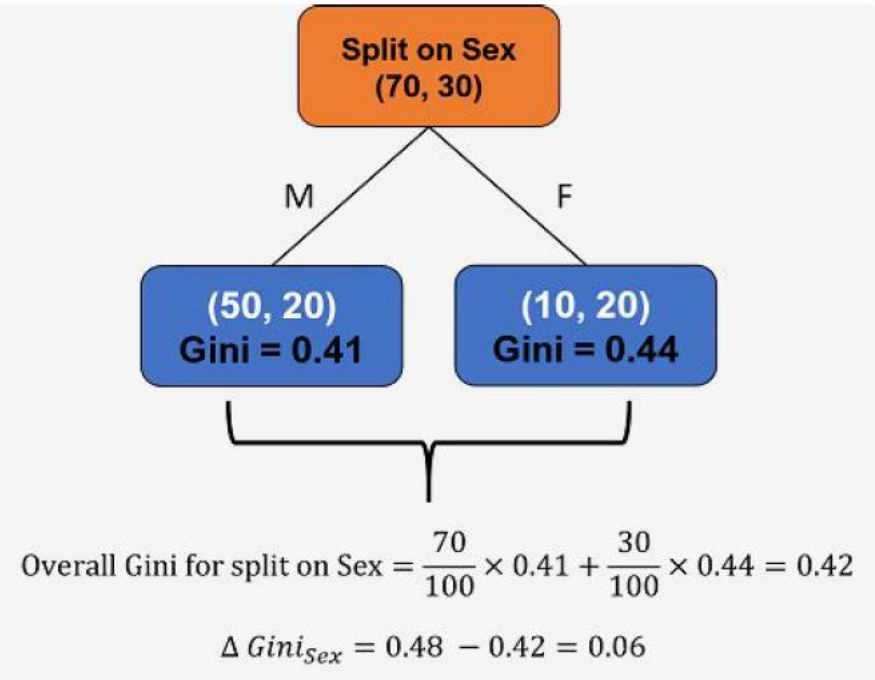
Now, you can calculate the gini index for the data before making any splits as follows:

Gini Impurity before split:

$$p_0(1-p_0) + p_1(1-p_1) = 0.6(1-0.6) + 0.4(1-0.4) = 0.48$$

Thus, the split based on gender gives the following insights:

- Gini impurity before split = 0.48
- Gini impurity after split = 0.42
- **Reduction in Gini impurity = 0.48 - 0.42 = 0.06**



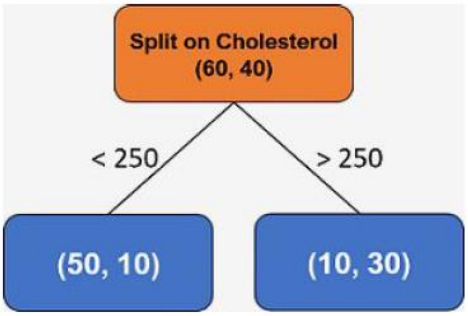

$$\text{Overall Gini for split on Sex} = \frac{70}{100} \times 0.41 + \frac{30}{100} \times 0.44 = 0.42$$

$$\Delta\, Gini_{Sex} = 0.48 - 0.42 = 0.06$$

**Split based on Cholesterol**

Let's now take another candidate split based on cholesterol. You divide the dataset into two subsets: Low Cholesterol (Cholesterol < 250) and High Cholesterol (Cholesterol > 250). There are 60 people belonging to the low cholesterol group and 40 people belonging to the high cholesterol group.

If you see the second table given above, you will notice that among the 60 low cholesterol people, 50 belong to class 0, i.e, they do not have a heart disease and the rest 10 belong to class 1 having a heart disease. So basically for the split on "Cholesterol", you have something like this —




Now the probabilities of the two classes within the low cholesterol subset comes out to be:

$$p0=50/60=0.833 \quad \text{and} \quad p1=10/60=0.167$$

Now using the formula, Gini impurity for low cholesterol subset becomes:

$$0.833(1-0.833)+0.167(1-0.167)\approx 0.27$$

Let's now take the other case where there are 40 high cholesterol (Cholesterol > 250) people out of which 10 belong to class 0 having no heart disease and 30 belong to class 1 having a heart disease. The probabilities of the two classes within the high cholesterol subset comes out to be:

$$p0=10/40=0.25 \quad \text{and} \quad p1=30/40=0.75$$

Now using the formula, Gini impurity for high cholesterol subset becomes:

$$0.25(1-0.25)+0.75(1-0.75)\approx 0.37$$

The overall impurity for the data after the split based on cholesterol can be computed by taking a weighted average of the impurities of the high and low cholesterol nodes. So, you have -

$$p_{low-cholesterol}=60/100=0.6 \quad \text{and} \quad p_{high-cholesterol}=40/100=0.4$$

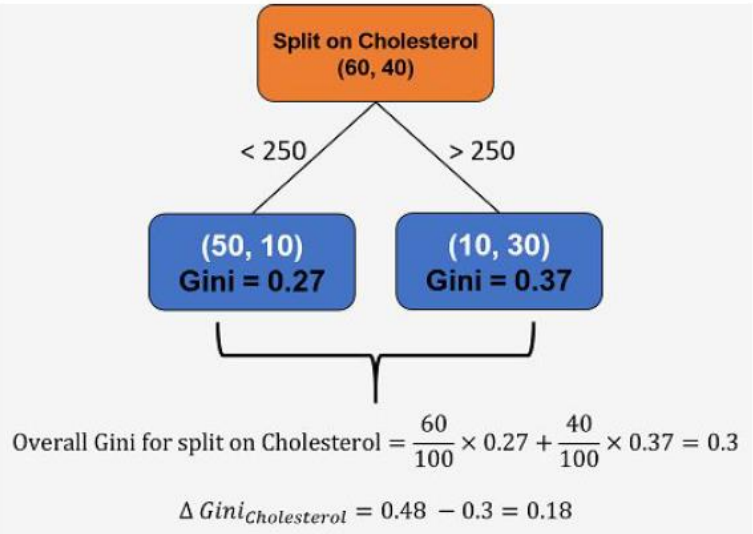
This gives the Gini impurity after the split based on cholesterol as:

$$0.6\times 0.27+0.4\times 0.37\approx 0.3$$

Thus, the split based on cholesterol gives the following insights:

- Gini impurity before split = 0.48
- Gini impurity after split = 0.3
- Reduction in Gini impurity = 0.48 - 0.3 = 0.18

Hence, you get the following tree after splitting on 'Cholesterol' —




1. **Truncation**: This process involves stopping the tree while it is still growing so that it does not end up with leaves containing only a few data points. Truncation is also referred to as pre-pruning.

2. **Pruning:** This process involves letting the tree grow to any level of complexity and then cutting the branches of the tree in a bottom-up fashion, starting from the leaves. It is recommended that you use pruning strategies to avoid overfitting in practical implementations.

- **criterion (Gini/IG or entropy):** This defines the function to measure the quality of a split. The sklearn library supports the 'gini' criterion for the Gini Index and the 'entropy' criterion for the information gain. By default, it takes the 'gini' value.

- **max_features:** This defines the number of features to be considered while finding the best split. You can input integer, float, string and None values.
    1. If an integer is an input type, then it considers that value as max features at each split.
    2. If a float value is taken, then max_features is a fraction and int(max_features * n_features) features are considered at each split.
    3. If 'auto' or 'sqrt' is taken, then max_features=sqrt(n_features) is considered at each split.
    4. If 'log2' is taken, then max_features= log2(n_features) is considered at each split.
    5. If the 'None' value is taken, then max_features=n_features is considered at each split. By default, it takes the 'None' value.

- **max_depth:** This denotes the maximum depth of a tree. It can take any integer or the ''None'' value. If it takes the ''None'' value, then its nodes are expanded until all the leaves are pure or contain less than min_samples_split samples. By default, it takes the 'None' value.

- **min_samples_split:** This denotes the minimum number of samples that are required to split an internal node. If an integer value is taken, then min_samples_split is considered to be the minimum number. If a float value is taken, then it shows the percentage. By default, it takes the '2' value.

- **min_samples_leaf:** This denotes the minimum number of samples that are required to be at a leaf node. If an integer value is taken, then min_samples_leaf is considered to be the minimum number. If a float value is taken, then it shows the percentage value. By default, it takes the '1' value.

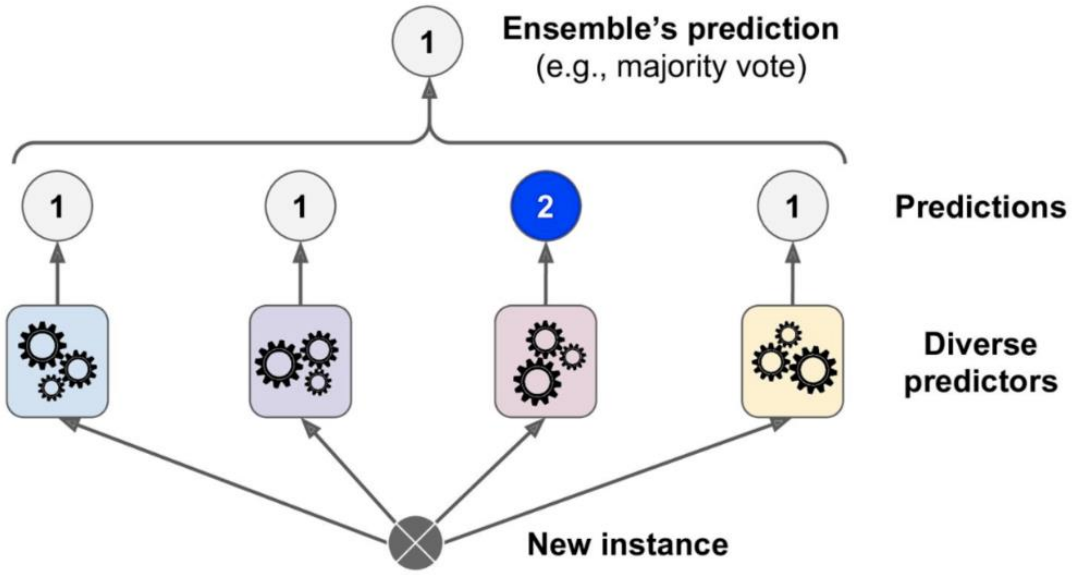
## Ensemble :

An ensemble means a group of things viewed as a whole rather than individually. In ensembles, a **collection of models** is used to make predictions, rather than individual models. Arguably, the most popular in the family of ensemble models is the random forest: an ensemble made by the **combination of a large number of decision trees.**

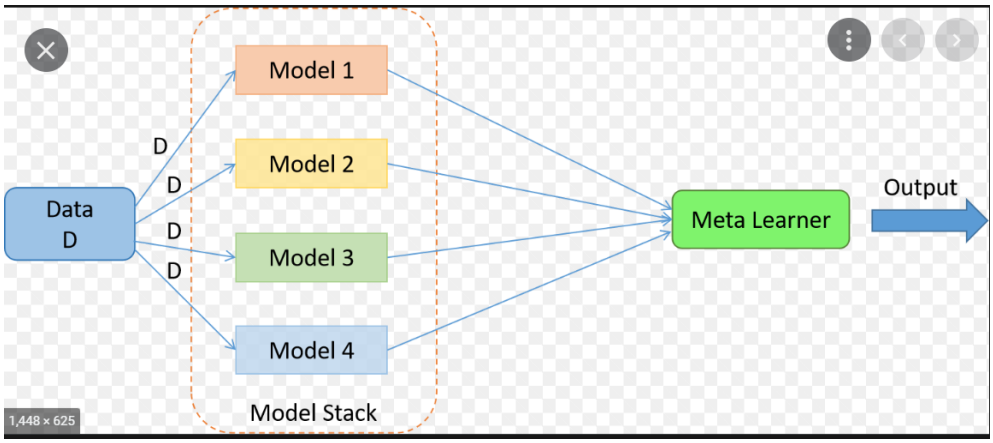
| Case | Result of Each Model | | | Result of the Ensemble | Probability |
|---|---|---|---|---|---|
| | m1 | m2 | m3 | | |
| 1 | Correct | Correct | Correct | **Correct** | 0.7*0.7*0.7 = 0.343 |
| 2 | Correct | Correct | Incorrect | **Correct** | 0.7*0.7*0.3 = 0.147 |
| 3 | Correct | Incorrect | Correct | **Correct** | 0.7*0.3*0.7 = 0.147 |
| 4 | Incorrect | Correct | Correct | **Correct** | 0.3*0.7*0.7 = 0.147 |
| 5 | Incorrect | Incorrect | Correct | **Incorrect** | 0.3*0.3*0.7 = 0.063 |
| 6 | Incorrect | Correct | Incorrect | **Incorrect** | 0.3*0.7*0.3 = 0.063 |
| 7 | Correct | Incorrect | Incorrect | **Incorrect** | 0.7*0.3*0.3 = 0.063 |
| 8 | Incorrect | Incorrect | Incorrect | **Incorrect** | 0.3*0.3*0.3 = 0.027 |

Now that you have a good understanding of what ensemble models are, let's look at some of the popular approaches to ensembling, such as voting, stacking, blending, boosting and bagging.

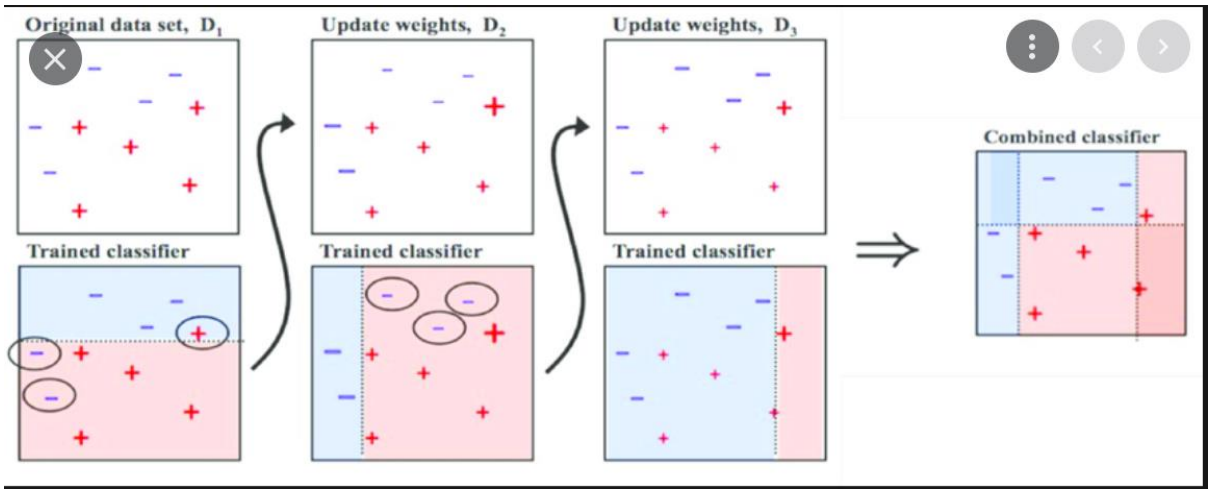
Voting :



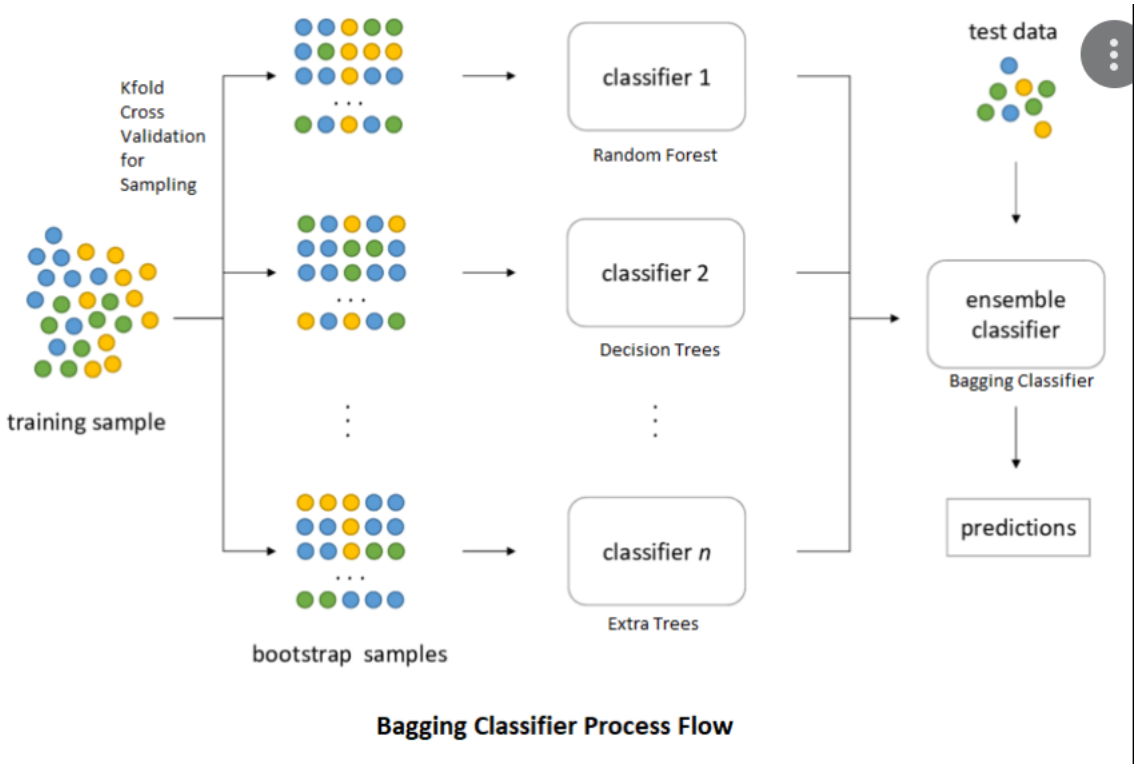
Stacking :




Boosting :




Bagging :




**Bagging Classifier Process Flow**

Random Forest:

**Random forests** are created using a special ensemble method called **bagging**. Bagging stands for **Bootstrap Aggregation. Bootstrapping** means creating bootstrap samples from a given data set. A bootstrap sample is created by **sampling** the given data set **uniformly** and **with replacement**. A bootstrap sample typically contains about 30-70% data from the data set. **Aggregation** implies combining the results of different models present in the ensemble.

OOB – Out Of Bag Error:

The OOB error is calculated by using each observation of the training set as a test observation. Since each tree is built on a bootstrap sample, each observation can used as a test observation by those trees which did not have it in their bootstrap sample. All these trees predict on this observation and you get an error for a single observation. The final OOB error is calculated by calculating the error on each observation and aggregating it.
It turns out that the OOB error is as good as **cross validation error**.

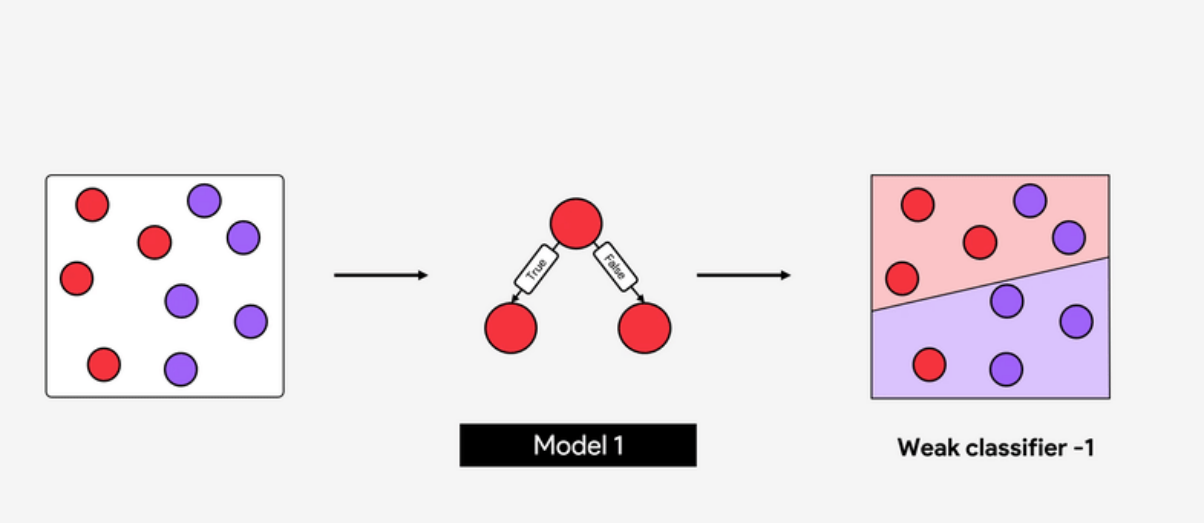
**Grid Search to Find Optimal Hyperparameters**

We can now find the optimal hyperparameters using GridSearchCV.

```
# Create the parameter grid based on the results of random search
param_grid = {
    'max_depth': [4,8,10],
    'min_samples_leaf': range(100, 400, 200),
    'min_samples_split': range(200, 500, 200),
    'n_estimators': [100,200, 300],
    'max_features': [5, 10]
}
```

Boosting:

Boosting was first introduced in 1997 by Freund and Schapire in the popular algorithm, AdaBoost. It was originally
designed for classification problems. Since its inception, many new boosting algorithms have been developed those
tackle regression problems also and have become famous as they are used in the top solutions of many Kaggle
competitions.
Let's start off with the basics of Boosting and move on to the boosting algorithms.
An ensemble is a collection of models which ideally should predict better than individual models. The key idea of
boosting is to create an ensemble which makes high errors only on the less frequent data points.
Boosting leverages the fact that we can build a series of models specifically targeted at the data points which have
been incorrectly predicted by the other models in the ensemble. If a series of models keep reducing the average
error, we will have an ensemble having extremely high accuracy.
Boosting is a way of generating a strong model from a weak learning algorithm.




ADA BOOS?

XG BOOS?

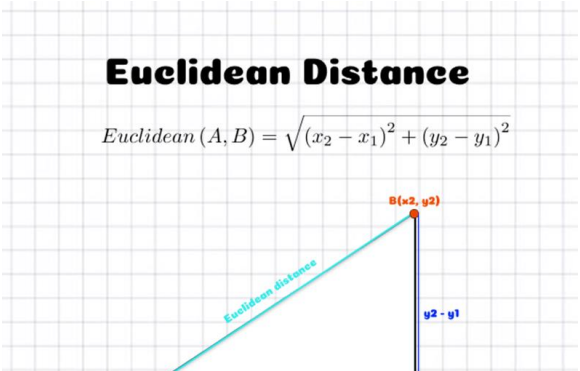
# Unsupervised Learning

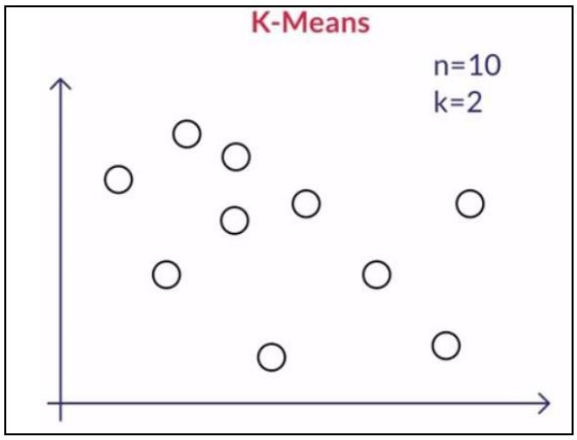
## Clustering:

- K-Means Clustering
- Hierarchical Clustering

K-Means clustering:

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.





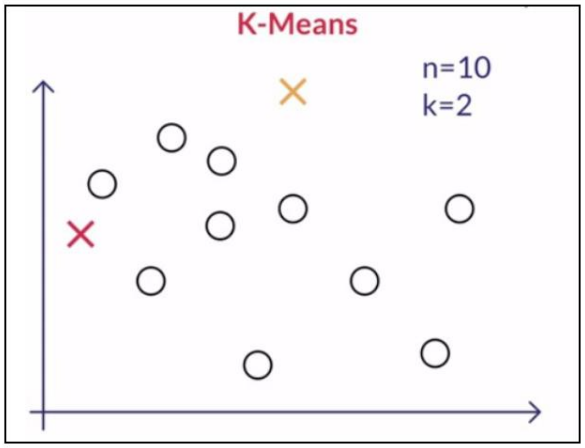
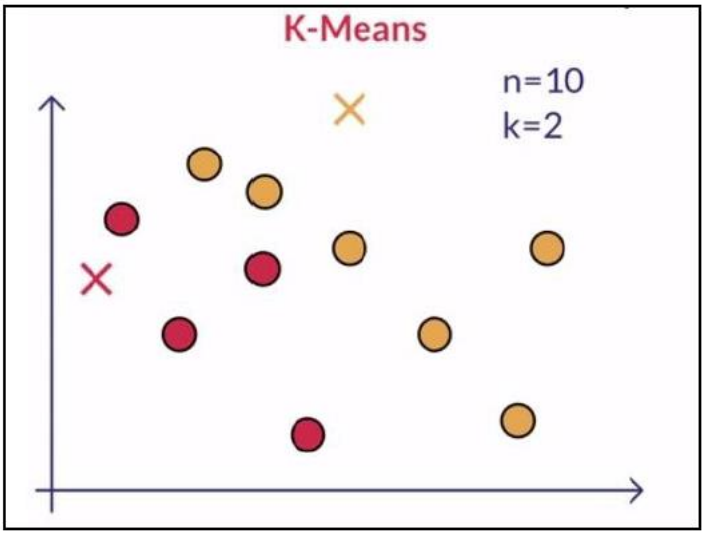
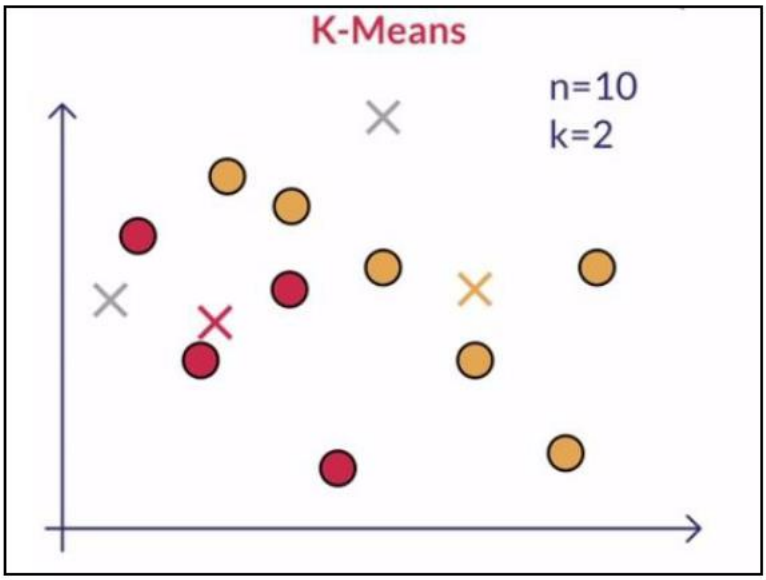
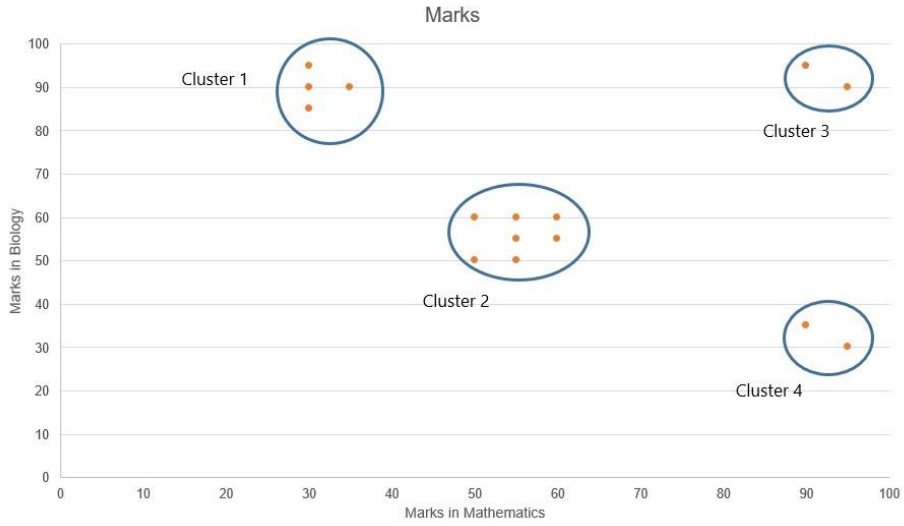




Fig 3: Choosing K random initial cluster centres

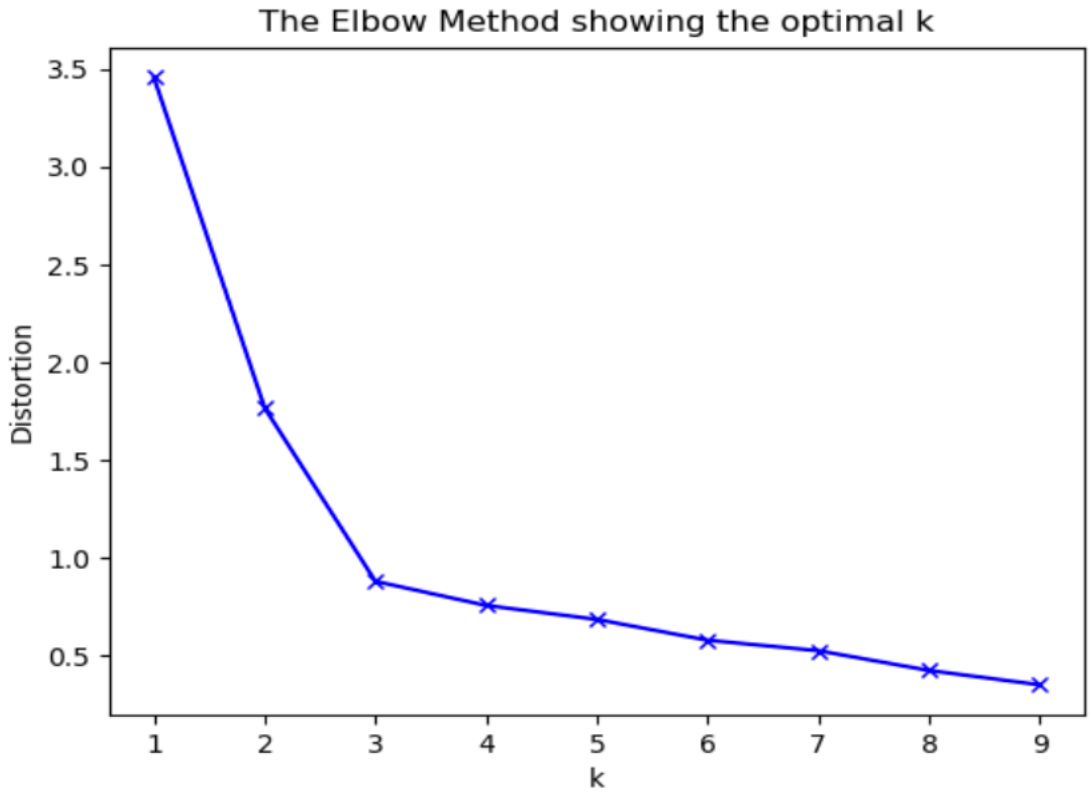





**Choosing the number of clusters K in advance?**

There are a number of pointers that can help us decide the K for our K-means algorithm:-
1. Elbow method:-
• Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance,
by varying k from 1 to 10 clusters.
• For each k, calculate the total within-cluster sum of square (wss).
• Plot the curve of wss according to the number of clusters k.
• The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.



3.) **Impact of outliers**

Since, the K-Means algorithm tries to allocate each of the data point to one of the clusters, outliers have serious impact on the performance of the algorithm and prevent optimal clustering.

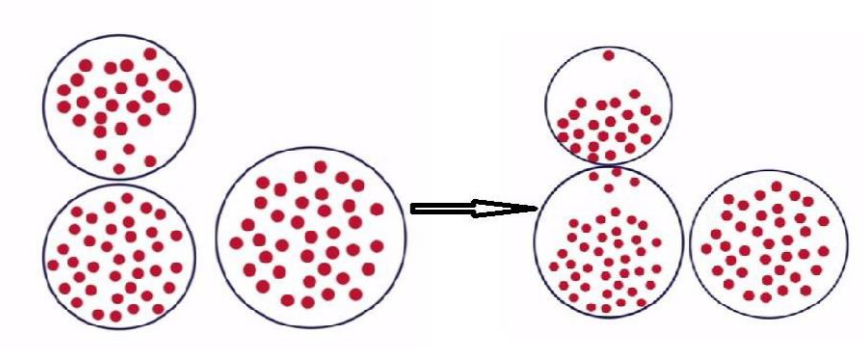
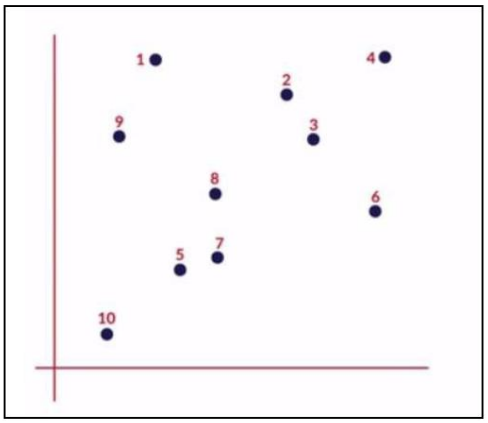
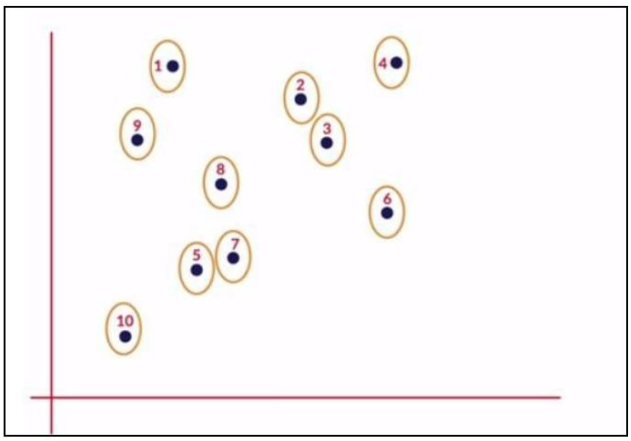


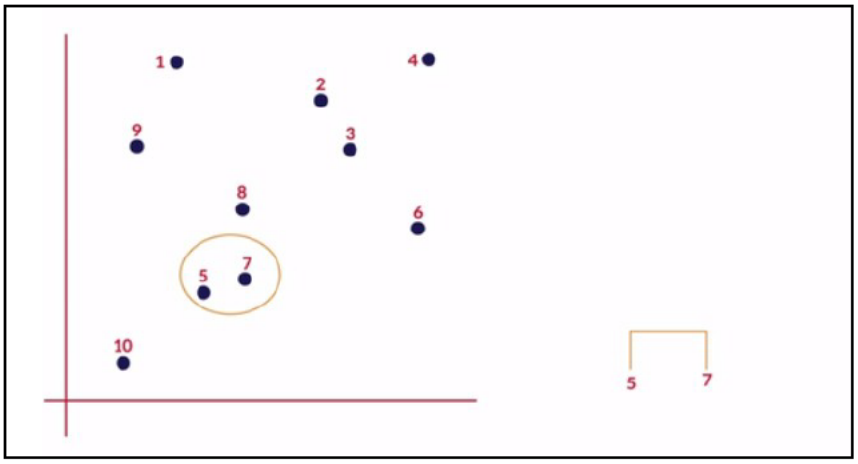
Fig 8: Impact of outliers on clustering

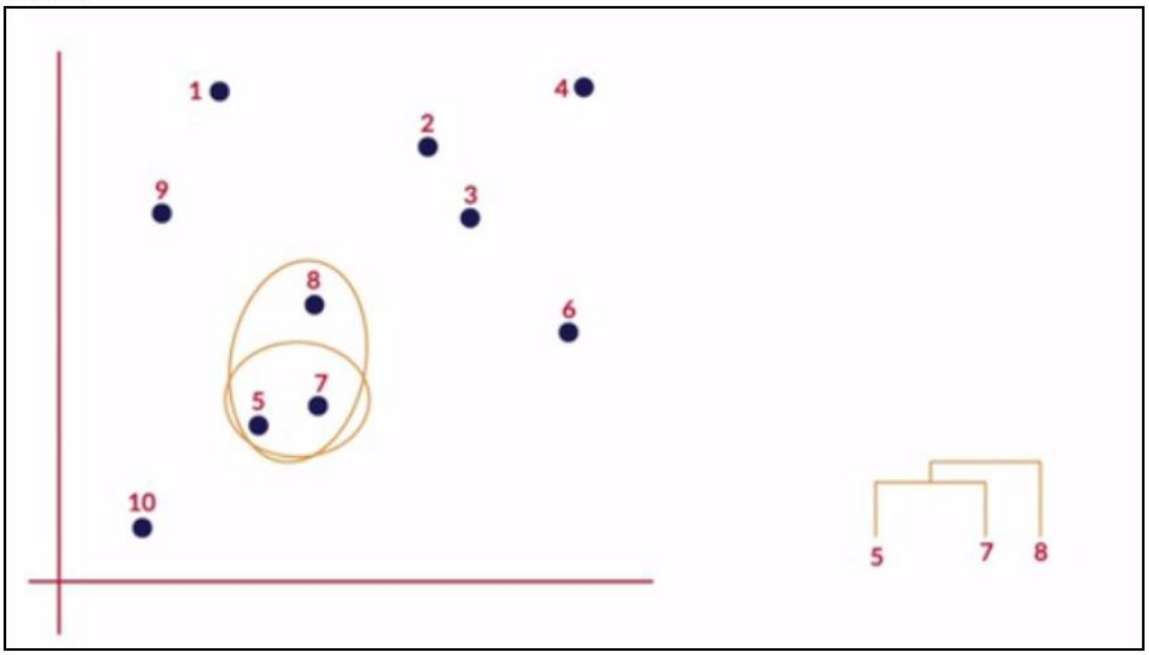
Hierarchical Clustering:

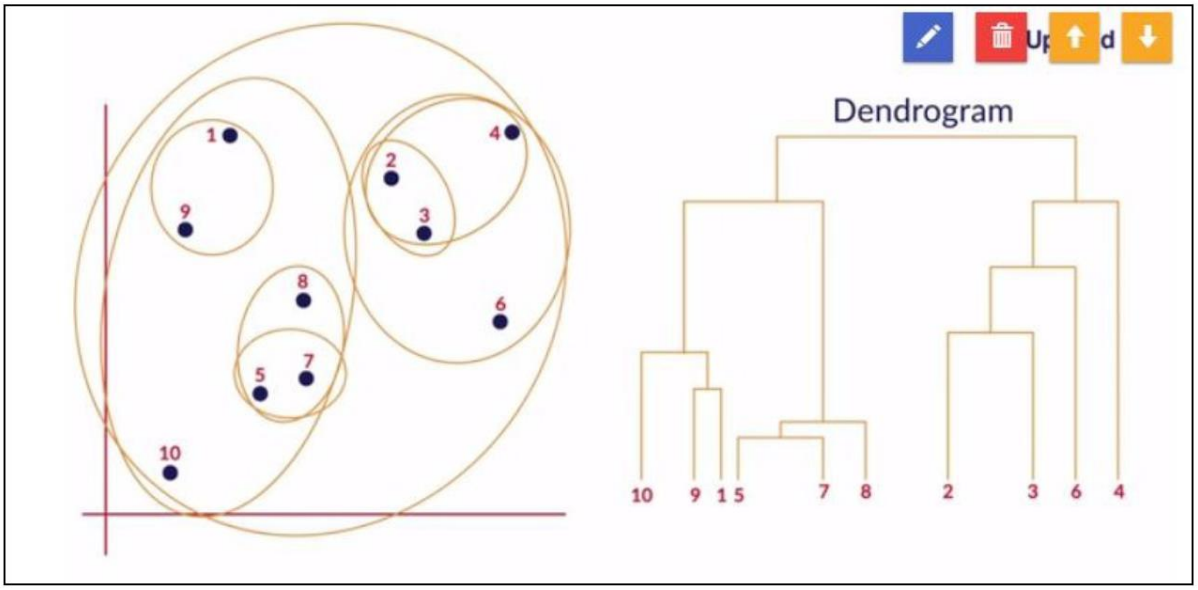
Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

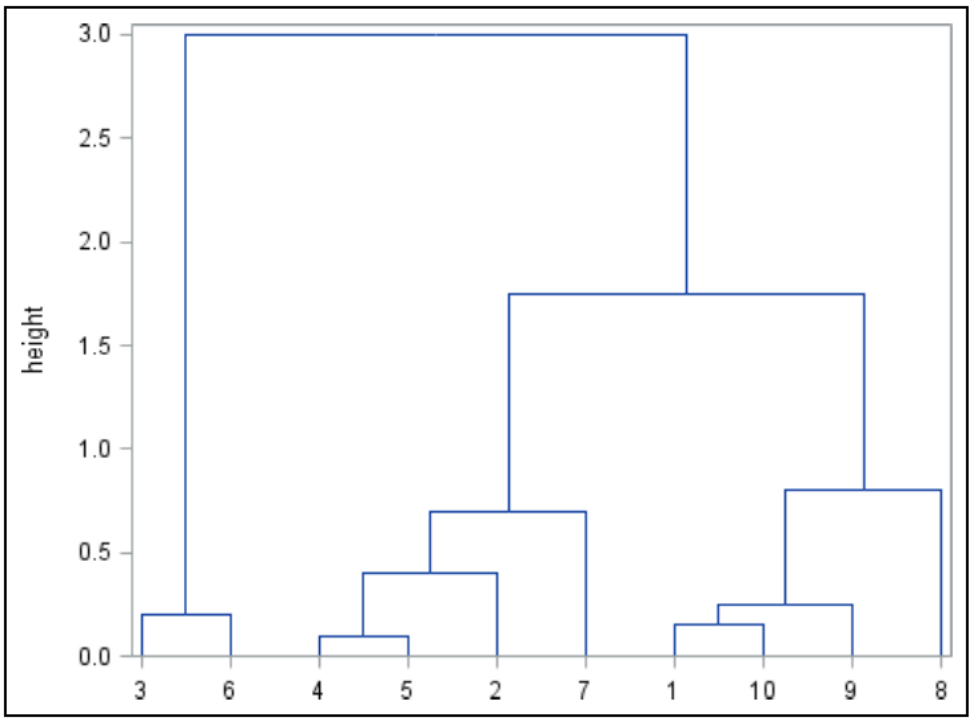


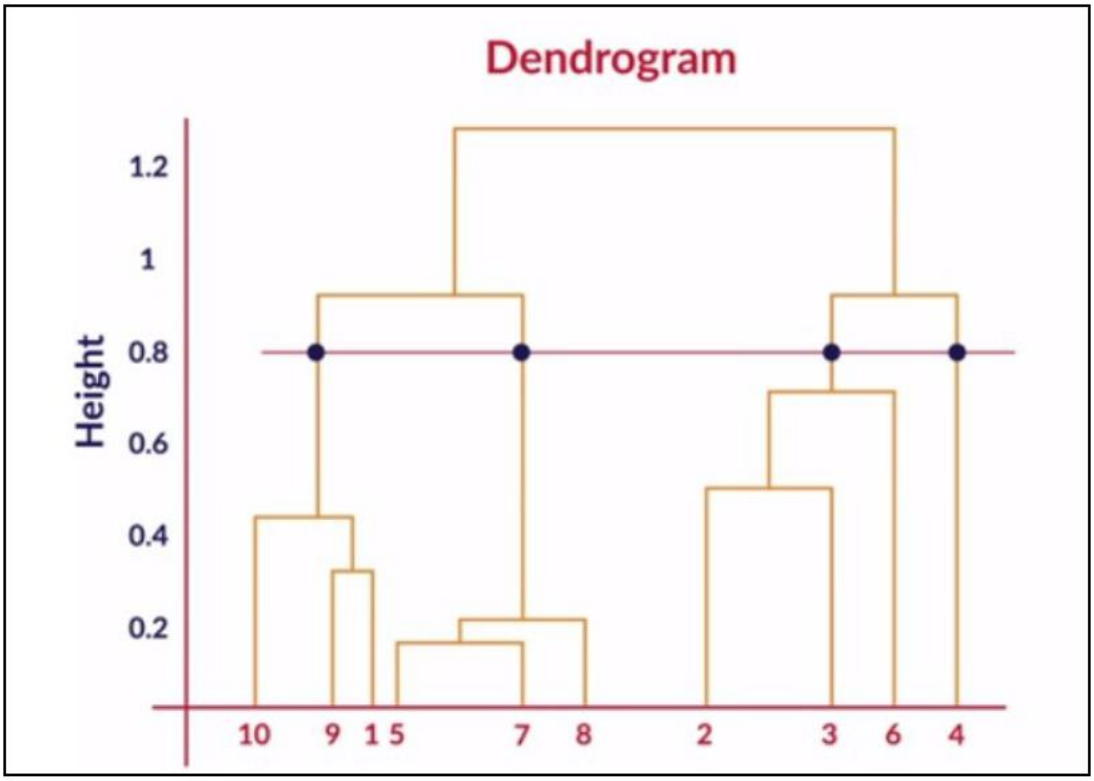

1
2
3
4
5
6
7
8
9
10

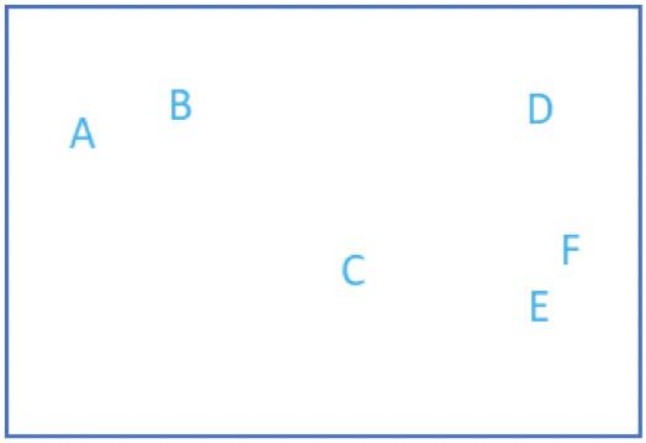

1
2
3
4
5
6
7
8
9
10
5
7

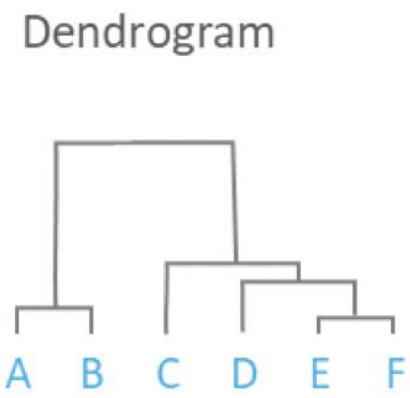

1
2
3
4
5
6
7
8
9
10
5
7
8


Dendrogram
1
9
2
3
4
6
8
5
7
10
10 9 1 5 7 8 2 3 6 4


height
3.0
2.5
2.0
1.5
1.0
0.5
0.0
3 6 4 5 2 7 1 10 9 8

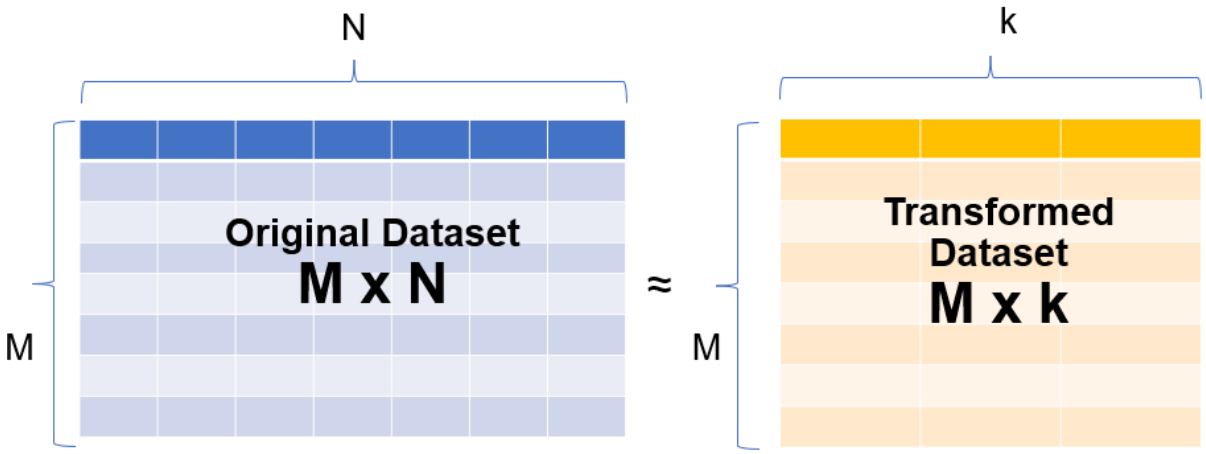

Dendrogram
Height
1.2
1
0.8
0.6
0.4
0.2
10
9
1
5
7
8
2
3
6
4


A
B
C
D
E
F


Dendrogram
A
B
C
D
E
F

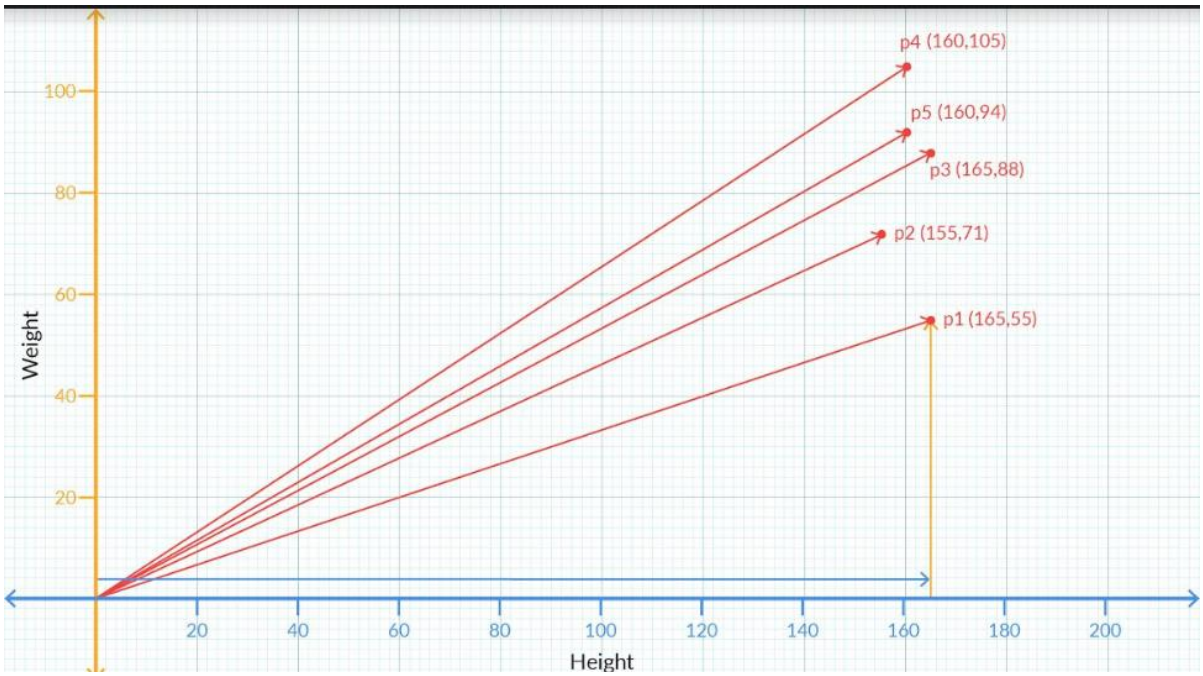
PCA

PCA is fundamentally a **dimensionality reduction technique**; it helps in manipulating a data set to one with fewer variables.




| Patient ID | Height (cm) | Weight (kg) |
|---|---|---|
| P1 | 165 | 55 |
| P2 | 155 | 71 |
| P3 | 165 | 88 |
| P4 | 160 | 105 |
| P5 | 160 | 94 |

$$\begin{bmatrix} 165 & 55 \\ 155 & 71 \\ 165 & 88 \\ 160 & 105 \\ 160 & 94 \end{bmatrix}$$



Reduction in variable size

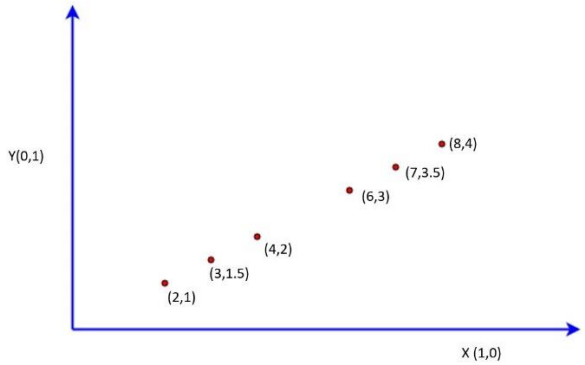
| Patient ID | Height (cm) | Weight (kg) |
|---|---|---|
| p1 | 165 | 55 |
| p2 | 155 | 71 |
| p3 | 165 | 88 |
| p4 | 160 | 105 |
| p5 | 160 | 94 |

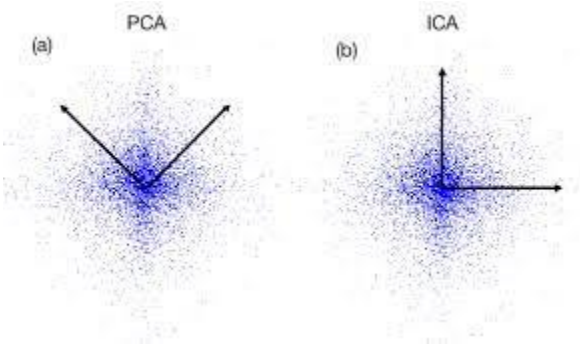


| Patient ID | Height (ft) | Weight (lbs) |
|---|---|---|
| p1 | 5.4 | 121.3 |
| p2 | 5.1 | 156.5 |
| p3 | 5.4 | 194.0 |
| p4 | 5.2 | 231.5 |
| p5 | 5.2 | 207.2 |

Basis $\left\{ \begin{bmatrix} 1cm \\ 0\ kg \end{bmatrix}, \begin{bmatrix} 0cm \\ 1\ kg \end{bmatrix} \right\}$ $\left\{ \begin{bmatrix} 1ft \\ 0\ lbs \end{bmatrix}, \begin{bmatrix} 0ft \\ 1\ lbs \end{bmatrix} \right\}$

Change of basis:



## Handson:

## EDA:

House_Price_Predicti
on_Project.ipynb

Bengaluru_House_Da
ta.csv

## Linear Regression:

Simple+Linear+Regr
ession+in+Python.ipy

advertising.csv

## Clustering:

Clustering+Python+L
ab.ipynb

Clustering_activity_K_Means.xlsx

## PCA:

PCA Demo Lipynb

Iris.csv