1 Basic Concepts:

- Data Science:

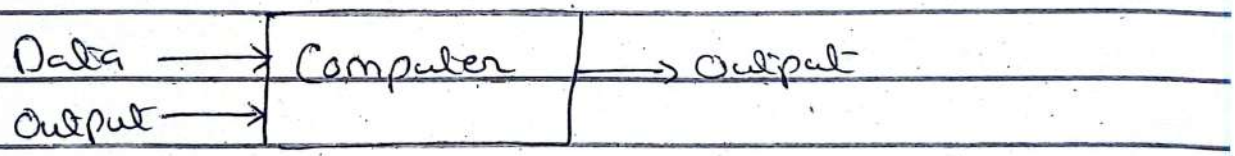
⇒ field of computer science that focuses on extracting meaningful insights, patterns and knowledge from data.

⇒ Consists of 5 stages:-

1) Data entry, extraction, acquisition.
2) Data cleansing, staging, processing, architecture
3) Data mining, classification, modelling, summarization
4) Predictive analysis, regression, text mining
5) Data visualization, reporting, decision making.

- Machine learning:

It is the idea that computer can learn from experience and examples.



It is like getting computers to program themselves
ML is automating the process of automation.

- ML Concepts and Algorithm:

Every ML algo has 3 components:

1) Representation of knowledge. → decision trees, set of rules, graphical models, neural networks and others
2) Way to evaluate programs → accuracy, prediction, posterior probability etc. [hypothesis]
3) Optimization of those candidate programs.

- Supervised Learning:

Data includes desired inputs and outputs. A model is prepared and through training process in which it is required to make predictions and is corrected when those predictions are wrong. this training process continues untill model achieves desired outputs.

- Unsupervised Learning: Data doesnot include outputs, But it is a model prepared by deducing structures present in input data

- Semi - Supervised:

Some desired outputs are provided but still model must learn the structures to organize data and make predictions.

- Deep learning:

is a ML technique that teaches computers to do what comes naturally to humans It's similar to the neurons in the human brains. It solves complex problems even when the data set is diverse, unstructured and inter-connected.

[2] Python Concepts:

- Lists:

my_list = [1, 2, 'saim', 4, 'Ali'].

↓

-> It's a list that can have any type of data in it and can be accessed using indexing and slicing, etc.

-> It's mutable -> can change its elements after its creation.

- Tuples:

my_tuple = (1, 2, 3, 'saim', 4)

↓

-> Tuples are same as lists but are immutable.

-> Have two functions:

1) Count -> counts the repetition of certain data in the tuple.

2) Index -> tell the index position of the first occurance of the desired/mentioned data.

- Dictionary: • (mutable).

• Uses keys instead of indexing.

my_Dictionary = {1: 314, 2: "saim", 7: "ali"}.

key data.

• Functions:

-> print(my_Dictionary.get(1))

used to get data at key '1'

-> .keys() = gives all the keys for a particular dictionary.

→ • clear() = empties the dictionary.

→ • copy() = copies one dictionary values (with keys) to another dictionary

→ • fromkey() =

e.g ⇒ a = { 1 : 5 , 2 : 10 , 3 : 15 }.

list_keys = [1 , 2].

b = a.fromkeys(list_keys, "saim")

new dictionary will have 2 keys of 1, 2 from 'a' dictionary but both will have value of ⇒ "saim"

→ • items() ⇒ shows a key and its value as an item and all the items are returned

– Sets =

• Special case of lists where elements can only be unique.

• Can't be indexed.

• Are mutable.

⁙ My_set = {3, 1, 2, 3} → no keys.

⁙ print(My_sets) → prints only unique values and no repeated ones.

-Functions =

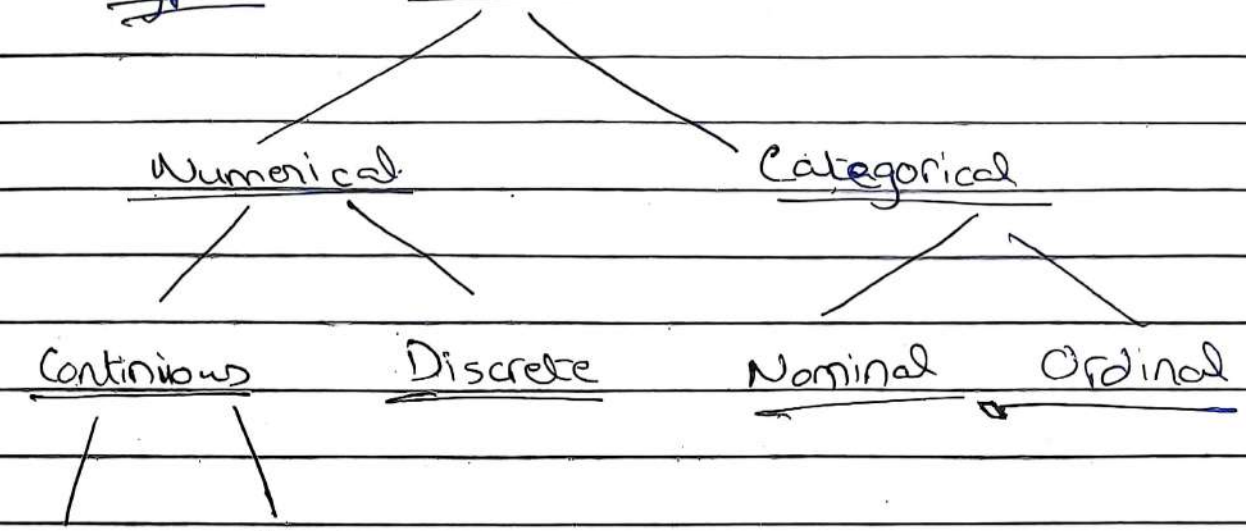
→ • add() = adds a data at the end of set.

→ • pop() = removes the last data

→ • discard() = removes a specific data from set. If that data is not in the set then it does nothing.

3 Statistics for D.S =

- Types = Data



- Outliers = extreme value in data set

[55, 88, 89, 90]

↓
outlier.

- Mean = Average of data. → but greatly affected by outliers.

$$\frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

- Median = Middle value

$$\left[\frac{n+1}{2}\right]^{th} \text{value}$$

- Mode = Most occured value in dataset.

- Variance =

$$\sigma^2 = \frac{\Sigma (x_i - \bar{x})^2}{n-1}$$

∴ x_i = i^{th} data point

$\bar{x}$ = mean.

n = no. of datapoint. intotal.

- Standard Deviation:

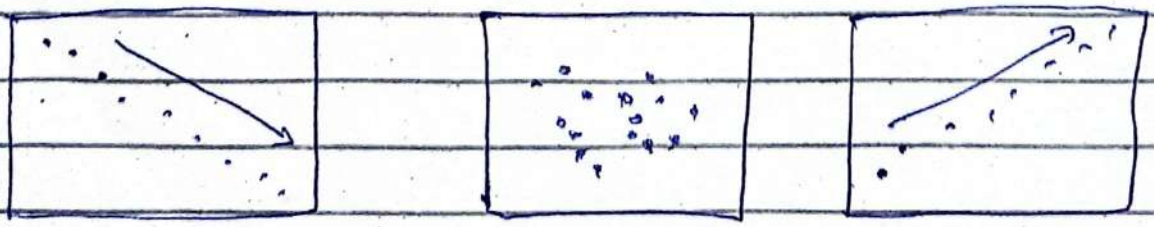
$$S.D = \sqrt{\sigma^2} = \sqrt{\left(\frac{\Sigma (x_i - \bar{x})^2}{n-1}\right)}$$

- Coefficient of Variation=

$$CV = \frac{\sigma}{\mu} = \frac{S.D}{mean}$$

- Covariance = measures relationship b/w 2variables

- Positive COV = If one var. increases, 2nd also increases.
- Negative COV = If one var. increases, 2nd var. decreases.
- Zero COV = no relation.



Negative COV Zero COV Positive COV

- Formula=

$$Cov(x,y) = \frac{\Sigma[(x_i - \bar{x})(y_i - \bar{y})]}{n-1}$$

($\bar{x}$: mean of x, $\bar{y}$: mean of y)

- Correlation= tells us strength/type of relationship that 2 var. have.

$$\text{Correlation} = \frac{Cov(x,y)}{\sigma x * \sigma y}$$

(σx: S.D of x, σy: S.D of y)

- Inferential Statistics=

-> Distribution= It is a function that shows possible variable for a variable and the frequency of their occurance

- Discrete Distribution → Uniform Dist. (Single Dice Prob).
- Discrete Distribution → Binomial Dist. (Double Dice Prob.)
- Continuous Distribution → Normal Dist

- have mean = mode = median.
- $N \sim (\mu, \sigma^2)$
  - N: Normal Distribution
  - μ: mean
  - σ^2: Variance

• Standard Normal Distribution:

mean = 0

S.D = 1

$$z = \frac{x - \mu}{\sigma}$$

z → z-score

- Central Limit Theorem:

Regardless of initial shape of population Dist., sampling Dist. will approximate to a normal Dist. As the sample size increases, samplens dist. will get narrower and and more normal.

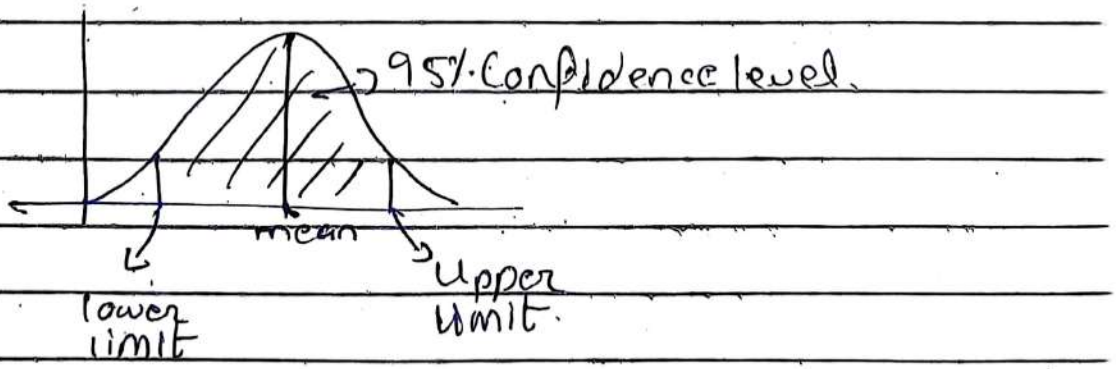
⇒ CLT ⇒

$$\mu_{\bar{x}} = \mu$$

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$ → Standard error.

• sample size ≥ 30.

- Confidence Intervals:



4 Probability:

$$P(x) = \frac{\text{Preferred outcome}}{\text{Sample Space}}$$

- Expected Values: $P(x) = \frac{\text{Successful Trials}}{\text{All Trials.}}$

- Relative Frequency:
It is the number of times the event occurs divided by Total no. of trials.

$$R.F = \frac{\text{Number of Success}}{\text{Total trials}}$$

- Theoretical Probability: a prob. that is expected like head has 50% chances and same goes for tails; when tossing the coin.

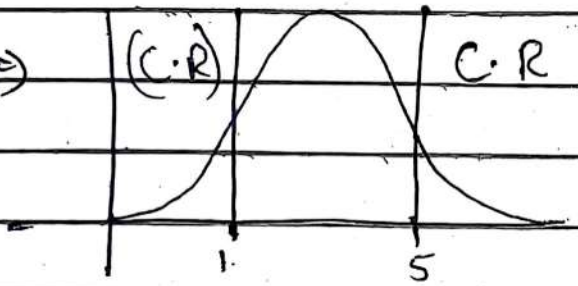
- Hypothesis testing:
  1) Make assumptions.
  2) Take initial Position (H_0).
  3) Determine alternate position (H_a)
  4) Set acceptance criteria.
  5) Conduct fact based tests.
  6) Evaluate Results
  7) Reach one of the conclusion:
     - favour alternate position
     - fail to reject initial position.

- Ho → Null Hypothesis.
- Ha → Alternate Hypothesis.

- Two-Tailed Test → (Ha) has > and < both. (Ha) has both directions.
- One-Tailed Test → (Ha) has either > or < (Ha) has one direction

- Critical Region: (C.R)

e.g.)



- Two-tailed test
- Ha → $1 < x < 5$

5 Exploratory Data Analysis

→ Unravel the data
→ To clean data
→ For feature engineering.

- **Process =** (Should be Quick efficient, Decisive)

- No. of Observations (rows)
- No. of features (columns)
- Data Type. (to identify type ML Problem)
- Problem type identification (find/look for outliers)
- Features that can be engineered.

- **Find out Data Types in set =**
in three groups.

① Numerical ② Categorical ③ Not sure

- **Plot numerical Distributions:**

- Histogram. → Used it to study Distributions
- look for: → outliers
→ Abnormal Distribution

- Plot Categorical Distributions:

- Use Bar Plots for it. -> for frequency of values.
- Box Plot as well.
- Use 2 features to identify relation b/w them.

- Study Correlations:

- look for:
  - strong relation b/w target variable and features.
  - study relationship of other features with each other as well.

6 Data Cleaning:

→ Feature Scaling:

(2 Types)

- Normalized data
- Standardized Data.

Fixing range of data that are b/w 2 values b/w (0 or 1) or (-1 and 0)

Tranform data to have mean → 0 and Standard deviation → 1.

- **Why do it:**

→ To bring all features to a similar range. so that no value dominates due to extreme values.

→ To improve model performance and training speed.

- **When do it:** (for algo's with rules)

→ When using ML Algos that calculate distance of data.

→ Neural Networks.

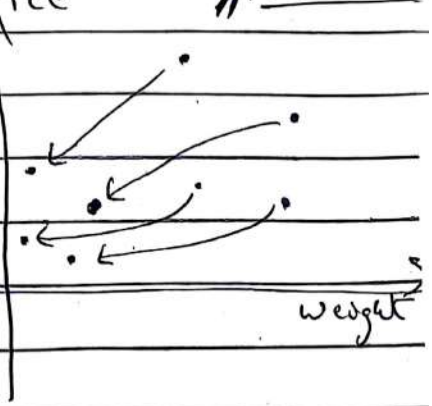
→ Knn

→ PCA.

- **Min, Max Scalar:**

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Values are shrinked



- Standard Scalar:

$$x_{new} = \frac{x - \mu}{\sigma}$$

→ Data Cleaning: (in Videos) → Go through the code as well.

→ Feature Engineering:

- Study of features.
- Combining, removing or adding features is engineering.

→ We can:

- rearrange sets for clarity
- Add additional colums for more info
- Add dumy variable. to convert categorical values into numerical form

7. Linear Regression:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- $\hat{y}$ → predicted value
- x_i is ith feature value.
- n is no. of features
- θ_j is jth model parameter or jth weights.

simplified: $\hat{y} = h_\theta(x) = \theta \cdot x$

- It is hypothesis fuction using model parameter θ.
- both are vectors.
- x → feature vector
- θ → parameter vector
- have dot product.

Cost Function:

- Mean Squared Error → MSE
- $$MSE(X, h_\theta) = \frac{1}{m} \cdot \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)}\right)^2$$
- It is sum of their square of all the (predicted values − actual values)
- Optimize weights.

- Gradient Descent:

- Optimization function.
- to optimize weights and minimize cost function.

→ Derivative of the cost function with each theta :-

$$\Rightarrow \frac{\partial}{\partial \theta_j} MSE(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)}\right) x_j^{(i)}$$

- Vectorization:

$$\nabla_\theta MSE(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} MSE(\theta) \\ \frac{\partial}{\partial \theta_1} MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\theta) \end{bmatrix} = \frac{2}{m} X^T (X\theta - y)$$

- Gradient Descent Step:

$$\Rightarrow \theta^{(\text{next step})} = \theta - \eta \nabla_\theta MSE(\theta)$$

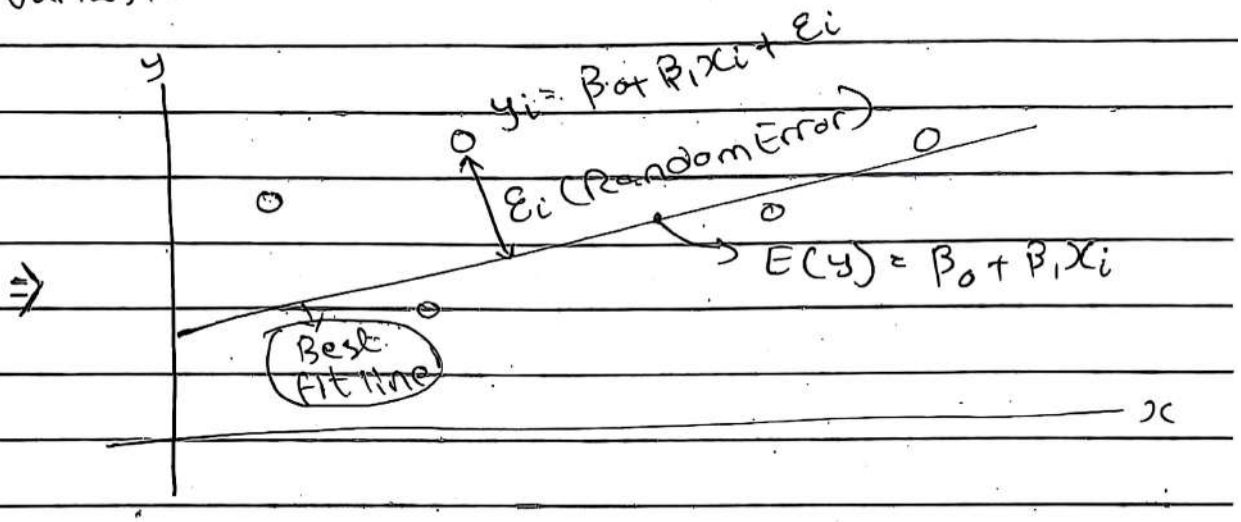
- It is learning Rate. (hyperparameter)
- tells size of Gradient Descent step.

- Linear Regression Model:

→ Tells relationship b/w Variables that is a linear function.

$$\Rightarrow \quad y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

- y_i: Dependent (Response) Variable
- β_0: Population y intercept
- β_1: Population Slope
- X_i: Independent (Explanatory) Variable.
- ε_i: Random Error



- Least Squares = (LS)

- It helps to get the best fit line.

$$\Rightarrow \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \hat{\varepsilon}_i^2$$

- LS minimizes the sum of squared differences (errors) (SSE)

✱ Coefficient Equations:

1) Prediction Equation:

$$\hat{y}_i = \hat{\beta}_0 - \hat{\beta}_1 x_i$$

2) Sample Slope:

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

3) Sample y-intercept:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}_i$$

⇒ ∴ Parameter Estimation Solution Table:

x_i	y_i	x_i^2	y_i^2	$x_i y_i$
⋮	⋮	⋮	⋮	⋮
$\sum x_i$	$\sum y_i$	$\sum x_i^2$	$\sum y_i^2$	$\sum x_i y_i$

$$\therefore \hat{\beta}_1 = \sum_{i=1}^{n} x_i y_i - \left[\frac{\left[\sum_{i=1}^{n} x_i\right]\left[\sum_{i=1}^{n} y_i\right]}{n} \right]$$

$$\sum_{i=1}^{n} (x_i)^2 - \left[\frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n} \right]$$

8 → Logistic Regression

(Prediction Model)

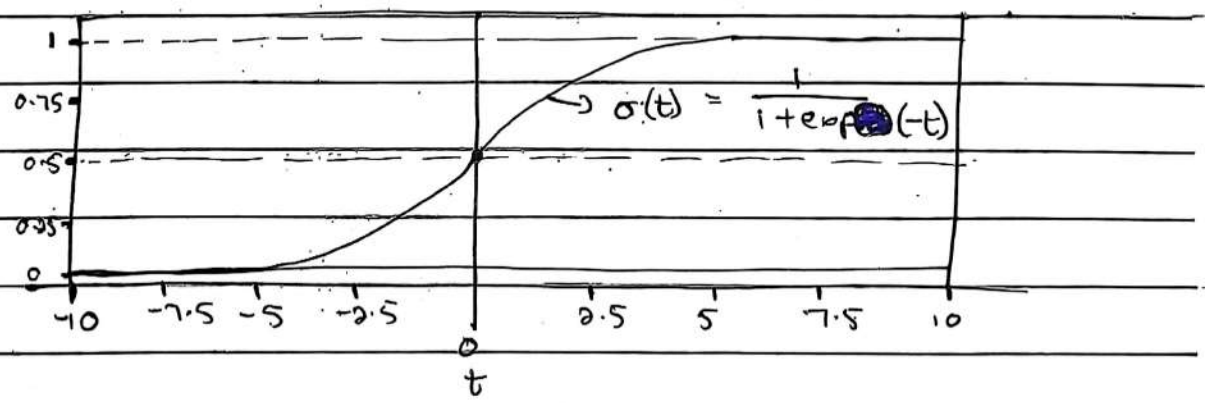
$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n.$$

$$\hat{p} = h_\theta(x) = \sigma(x^T \theta).$$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}.$$

Activation function.

- **Estimating Probabilities =**



$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

- **Cost function =**

→ Cost function of one training instance

$$\Rightarrow c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1. \\ -\log(1-\hat{p}) & \text{if } y = 0. \end{cases}$$

→ Cost function over the entire training instance

$$\Rightarrow J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{p}^{(i)}) + (1-y^{(i)}) \log(1-\hat{p}^{(i)}) \right]$$

• Optimization Algorithm=

→ Partial derivative with respect to all theta

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\sigma(\theta^T x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

<- - - - - - - - - ->

→ K-Nearest Neighbours = (KNN) 9

1) Parametric Models=

• Here we know which model exactly fits to the data like linear regression line.

$$y = w_1 x + e.$$

• Here parameters are W's.

- Regression looks like a linear line.

2) Non-Parametric Models=

• The data tells you what the regression should look like.

$$y = f(x) + e$$

(f(x) is decided by the data)

(It can be any function)

• Here no. of Parameters are decided by data

• f(x) can be perfectly approximated by infinite parameter model.

⇒ EDA on IRIS DataSet:-

- For KNN section, Iris dataset will be used.
- It has four Attributes:

① sepal length in cm.	∴ sepal is outmost part of flower, at the base of the petals.
② sepal width in cm.	
③ petal length in cm.	
④ petal width in cm.	

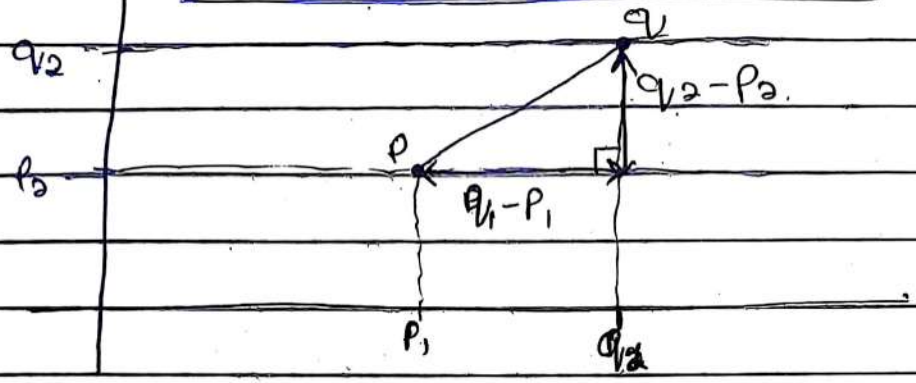
- For this course I will use 2 features for easier visualization :- sepal length and width.
- Class for the flower can be:-
  1. Iris setosa.
  2. Iris versicolor
  3. Iris virginica

  → [each has above attributes mentioned]

- KNN Algorithm Steps:-

(1) Calculate distance using eculidean distance:-

$$d(p,q)^2 = (q_1-p_1)^2 + (q_2-p_2)^2$$



(2) Find the nearest neighbour using the python code.

(3) Classify the point based on the majority vote.

(4) Combine all these three steps in one function to make prediction for each test point.

∴ Code for all these steps is done in the google colab.

→ Performing hyper-parameters tuning using K-Fold Cross Validation=

- Never use test data for tuning → It is only for final evaluation.
- Use a validation set (or cross-validation) to tune hyper parameters.
- Dataset is usually split into train/validation/test [(60%, 20%, 20%) → split]
- Cross-Validation: split data into k-groups; train on (K-1) groups; test on 1 group and repeat.
- This way every group gets used as a test set once.
- It helps in using full dataset effectively without wasting data.

→ Choosing the Distance Measurements=

- Use Euclidean distance if the dimensions are measuring similar properties (like width, height, depth)
- Use Manhattan distance if the dimensions are disimilar (like age, weight, gender).

→ Need for normalization of data when using KNN=

- KNN uses distance-based similarity.
- So features with larger ranges would otherwise dominate
- Normalization ensures balanced influence of all features.

10 Decision Trees:

→ We will do Exploratory Data Analysis on the Adult dataset

→ Objective is to find whether a person's income exceeds 50K $/per or not. From that we will decide if that person is eligible for credit card on not.

→ **Entropy:**

It is used to measure impurity or disorder in the data. Tells how uncertain or mixed the data is.

⇒ Entropy $(H(\pi)) = -\sum \pi \log_2(\pi)$.

→ **Information Gain: (IG)**

Condition Entropy $H(Y|X) = -\sum p(Y=y, X=x) \log \frac{P(Y=y, X=x)}{P(X=x)}$.

⇒ $IG = H(Y) - H(Y|X)$

[amount of uncertainty before] [Amount of uncertainty after]

- IG measures how much uncertainty (impurity) is reduced after a split.

→ The Decision Tree ID3 Algorithm=

- Steps=

① Find best feature
② Find best split for the best feature.
③ Partition classes based on step 1 and 2.

① - Partition Classes=

- We do this to separate data into smaller, purer groups, so that each group is more uniform.
- This reduces uncertainty, making decision rules simpler, and improves the accuracy of predictions.

② - Find the best Split=

- 'Find the best Split' function is made in the code that find best place to split the data (on a chosen attribute) when building a decision tree.

- We do this because:-
  ① Decision tree works by splitting data step by step into smaller groups.
  ② Good split makes group purer.
  ③ To measure how good a split is, we use information gain (how much uncertainty in reduced).
- Function tries all possible split values and picks the one with highest information gain

- How it works:-
  ① Pick an attribute. → e.g "Age"
  ② check all possible values of it (like 20, 30, 40).
  ③ SPLIT dataset into left, right for each value
  ④ Calc. Information gain and choose value that gives best Info. Gain

③ - Find the best feature:-

- What is a feature:-
  → It is an attribute or column that describes each data point.
- What does function 'find best feature' do:-
  → As we used this function in code, it looks at all the features and asks: "Which feature, if I split data on it, will give me best seperation b/w classes?"
- Why do this:-
  Because in the decision tree, first split is the most important.
  → If we choose wrong feature to split on, the tree will be messy and less accurate.

→ Important Hyper-Parameters in D.T:-

① Criterion:-

- It's a function used to find/measure how good a split is.
- Gini ⇒ default, measure impurity.
- Entropy ⇒ uses information gain
- log_loss ⇒ for probability.

② max_depth:-

- It's the max no. of levels (depth) the tree can grow
- It's to prevent tree from growing too deep and overfitting.

③ Class_Weight:=

- Assigns weight to classes (like in imbalanced data sets)
- It helps the model pay more attention to minor class.

④ splitter:=

- The strategy used to split nodes.
- 'best' is for choosing best split.
- 'random' is for adding randomness.
- Used for best random split to be choosen.

⑤ Min_Sample_Split:=

- Min. no. of samples required to split an internal node.

→ Gini VS Entropy:=

① Gini =

- Tells use how mixed up the classes are in a node
- If samples are from one class → Gini = 0 (pure)
- If they're evenly split → Gini = 1 (impure).
- Formula:-

$$Gini = 1 - \sum (p_i^2)$$

∴ p_i is proportion of samples belonging to class i.

- Range: 0 - 0.5 (for binary classification)
- Used cause it's faster to compute than entropy, work well in most cases.

② Entropy:=

- Measures amount of disorder/uncertainty in data.
- Higher Entropy = More disorder.

Usage:
- Provides more information-theory-based split.
- Slightly slower than Gini.
- Formula:-

$$-\sum (p_i \log_2(p_i)) = Entropy$$

→ Pruning =

- Pruning is a process of cutting down extra branches in a decision tree.
- To reduce overfitting and make tree more general and simpler.

- It is needed because D.T can become too deep and memorize training data (overfitting).
- Pruning removes unnecessary splits so the model can predict better on new data.

→ Types of Pruning:

(a) Pre-Pruning:

- Stop tree from growing too deep.
- Examples:
  - max_depth :- max-levels allowed
  - min_sample_split :- min-samples needed to split a node.
  - min_sample_leaf :- min-samples in leaf node

(b) Post-Pruning:

- First grow a large tree, then remove weak branches
- Uses a parameter called ccp_alpha (Complexity parameter).
- Higher ccp_alpha -> more pruning -> simpler tree.

→ How Is Post-Pruning Performed =

① Grow a full tree =

- First, allow the tree to grow very deep (almost no restrictions).
- This usually causes overfitting.

② Calculate "Cost-Complexity" for each subtree =

- For every possible branch (subtree), calculate :-

$$R_\alpha(T) = R(T) + \alpha \times (\text{number of leaves in } T)$$

∵ - $R(T)$: error of subtree.
- α : complexity penalty (ccp_alpha)
- more leaves : higher penalty.

③ Find weaker branches =

- Identify branches that don't reduce error much add complexity.
- These are the "weakest links"
- Note C.C helps prune unnecessary branches to avoid overfitting.
  ↳ It means that model is "too smart" on the training data but "too dumb on new data

④ Pruning them=

- Cut those weak branches (remove unnecessary splits).
- This reduces tree size without loosing much accuracy.

⑤ Repeat until optimal tree is found=

- Keep pruning step by step.
- Stop when pruning starts hurting accuracy.

→ Pros of Decision Tree=

- Easy to understand and interpret
- Can work directly on raw data hence no need for data scaling/normalization.
- Handle both numerical and categorical data
- Require little data preprocessing.
- Fast training and predictions.
- Show which attributes matter most.
- Can capture non-linear relationships unlike KNN

→ Cons of Decision Tree=

- Overfitting: deep trees memorize training data and fail to generalize.
- Sensitive to [noisy data] → [It can overfit it] then create D.T.
- Biased with imbalanced dataset, so balance it first.
- Small variation can result in different D.T.
- As branching goes deeper, you get exponentially less data.

III Ensemble Methods + Learning

- Ensemble learning is combining many mediocre (weak) models into a one supermodel.

- Accuracy = $1 - \prod_{i=1} \text{Error}(i)$

Product of errors of base learners.

∵ notes atleast on of the baselearner should be correct.

→ This formula has a problem that we are assuming all the predictions are completely independent of eachother which is unrealistic.

→ Solution to this is bagging and boosting.

- Bootstrap Sampling = It is a statistical technique where you repeatedly draw samples with replacement from the original dataset to create many new "boostrap samples" of the same size.

- Once we collect those samples, we calculate statistics on each bootstrap sample like mean, median, coefficient, regression etc
- We choose 'x' no. of samples from set and each of size 'n'. Then we perform statistic like mean for each and take average of all mean. Then decide or conclude that the avg. is the mean for whole population

- Bagging: → [Bootstrap Aggregation]

- Its basic idea is to create different models using different bootstrapped training sets and average their prediction to produce final output of model.
- It is good for:
  1) Reducing variance of model. (due to replacement)
  2) Reducing overfitting
  3) Outliers are likely omitted in some of training bootstrap samples.
- Steps of bagging are:-

1) Generate B different bootstrap training datasets.
2) Train our ML Method on the b bootstrapped training set.
3) Final prediction:

$$\hat{f}(x) = \frac{1}{B} \cdot \Sigma \hat{f}^b(x)$$

:- B = Total number of bootstrap models.
- $\hat{f}^b(x)$ = prediction from b^{th} model.
- Σ = sum over all B models

- Out-of-Bag Error (OOB) =

- OOB error is a way to estimate model performance in bagging without needing a separate validation set.
- When we create samples, there are high chances that some of the data points from original dataset are left out and not used in any of the samples. These are called 'out of bag' observations.
- So those observations are used as test set for the trained model; as they are unused and unseen by trained model.

- Random Forest =

- It is slightly modified bagging algorithm.
- It is basically collection of decorrelated D-Trees
- For every tree:
  1) we perform bootstrapping.
  2) For each bootstrap, we choose attributes at random.
  3) we pick features randomly on each split and built tree only on them with current subsample
- Classification is done by taking a majority vote of classifications yielded by each tree in the forest after it classifies an example

- Pros of Random Forest:

- Robust to outliers.
- Works well with non-linear data.
- Lower risk of overfitting.
- Runs efficiently on a larger dataset.

- Cons of Random Forest:

- biased while dealing with categorical variables.
- Slow Training if code is not optimized (just like our implementation took more than 3min. to execute the evaluation code, but for sklearn it took less than 10 seconds).
- Now suitable for linear method with alot of sparse features.
- They are greedy algorithms.

- Boosting:

- It is a technique where we combine many weak models to build one strong model.
- Working:

1) Train the first model (DT)
2) Focus on the mistakes. When next tree is trained, it gives more weight to misclassified point, so it tries more to fix those mistakes.
3) Repeat the process as each tree learns from previous one and gets more better at reducing errors
4) At last, predictions are combined and finalized using classification regression.

- AdaBoost - (Adaptive Boost):

- One of boosting algorithms.
- It builds many weak classifiers and combine them into a strong classifier by giving more weight to the difficult-to-classify points.

→ Working:

1) Starts with equal weights: Every training sample is given equal importance (weight)
2) Train the first weak learner: Fit a simple classifier (like decision stump) using these weights.
3) Check errors: see what point weak learner classified wrong.
4) Increase weight of misclassified points
5) Assign a weight to a learner
   - Better learners get higher weight.
   - Weaker learner still count, but less.
6) Repeat step 2-5.
7) Final Prediction = weighted majority vote (for classification) or weighted sum (for regression)

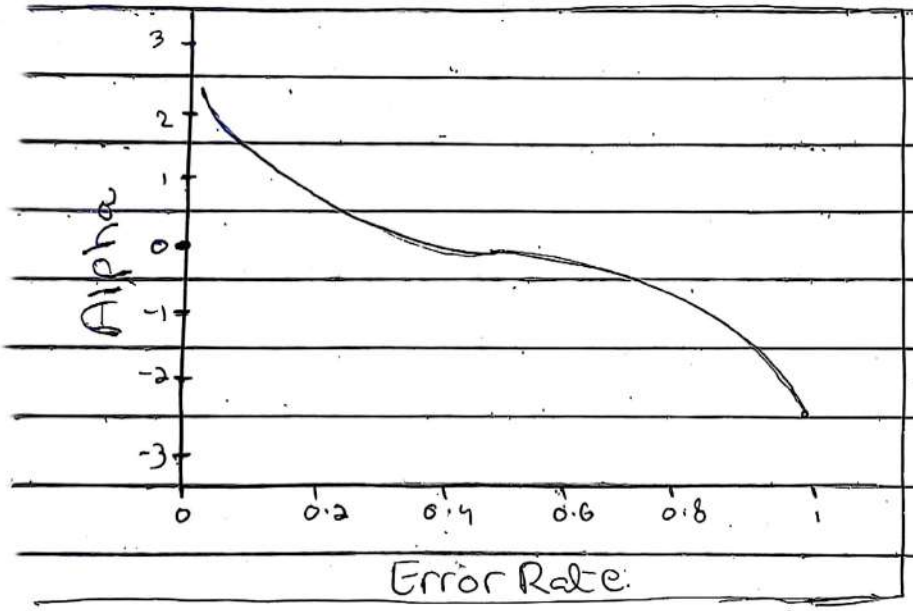
→ Decision Stumps:

- Are simplest form of decision trees.
- They have one split only (one level deep)
- Make decision based on just one feature and one threshold.
- AdaBoost combines many stumps to form one strong model

→ AdaBoost Equation:

$$\Rightarrow \alpha_t = \frac{1}{2} \ln\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

- α is how much influence this stump would have in the final classification.
- Total Error is nothing, but the total num. of misclassifications for that training set divided by the training set size.



weight of next iteration → $w_i = w_{i-1} \cdot e^{\pm \alpha}$ ← weight of previous iteration.

- If a sample perform well then the sample weight decreases from what it was before; in the next interation.
- It increases if it doesnot perform well

Just used to update weight of each sample after iterations.

→ AdaBoost Pseudocode :

- Initially set uniform example weights.
- For each base learner:
  1) Train base learner with a weighted sample.
  2) Test base learner on all data.
  3) Set learner weight with a weighted error.
  4) Set example weights based on ensemble predictions.

12 Support Vector Machine = (SVM)

- SVM =

- It is a supervised machine learning algorithm used for classification (mainly) and regression.
- It tries to find best boundary (hyperplane) that seperates different classes in the data.

- How it works =

1) Plot data :- Imagine you have points of two classes (red = cats, blue = dogs)
2) Find the seperating line :- SVM finds the line (in 2D), or (in 3D) plane, or hyperplane (in higher Dimensions) that best seperates the two classes.
3) SVM chooses the margin that leaves largest gap b/w classes.
4) Support vectors :- The point closest to the boundary are called support vectors as they define position

of the boundary.

5) For Non-linear Data :- If classes can't be seperated by a straight line, SVM uses a kernel trick to map data into higher dimension where seperation is possible

- Why to maximize this Margin?=

1) A larger margin reduces the risk of overfitting, making the model perform well on unseen data.
2) With a wide margin, small variations or noise in the data are less likely to cause misclassification.
3) Points farther from the boundary are classified with higher certainty, improving reliability.
4) Margins should be maximized inorder to prevent new data points from being misclassified.

- Hard vs Soft Margins=

→ Margin Equation:=

1) Decision boundary:-

- SVM tries to draw a line/plane to separate classes
- Line is given by ⇒

$$w \cdot x + b = 0$$

w = weight vector (define orientation of plane)
b = bias (shift the plane)
x = input vector

2) The Margins:-

- SVM creates two parallel boundaries on which the supporting vectors lie given by ⇒

$$w \cdot x^+ + b = +1 \rightarrow \text{(side of class +1)}$$
$$w \cdot x^- + b = -1 \rightarrow \text{(side of class -1)}$$

3) Margin Width:-

It is the distance b/w two boundaries

$$w \cdot x^{+} + b = +1$$

$$- w \cdot x^{-} + b = -1$$

$$\Rightarrow w(x^{+} - x^{-}) = 2.$$

∴ divide both sides by $\|w\|$.

∴ $\|w\|$ is the size of the weight vector, which controls how tilted line/plane is.

$$M \Rightarrow \frac{w(x^{+} - x^{-})}{\|w\|} = \frac{2}{\|w\|}$$

$$\Rightarrow \boxed{M = \frac{2}{\|w\|}} \quad \ldots\ldots$$

→ To Maximize M we want to minimize $\|w\|$ so optimized eq is:

$$\Rightarrow \phi(w) = \frac{1}{2} w^{T} \cdot w.$$

Subject to :-

$$y_i(w^{T} x_i + b) \geq +1 \quad \text{if } y_i = +1$$

and $$y_i(w^{T} x_i + b) \leq -1 \quad \text{if } y_i = -1$$

→ Hard Margin:

- It draws a line/hyperplane that perfectly seperates two classes
- No mistakes are allowed - every point must be on correct side
- It require complex kernel if some points donot fit within the boundary causing overfitting

→ Soft Margin:

- More flexible.
- Seperates the classes with biggest gap, but allows some points to be on wrong side.
- This makes it work better on real-world noisy -data and reduces overfitting.

Now we modify equation:-

$$\Rightarrow \boxed{\frac{1}{2} w^T \cdot w + C \sum \varepsilon_K}$$

∴ • ε_K → slack variable (define how much a point violated the margin)

controls overfitting. • C → penalty parameter that balances margin size vs errors.

Decrease Variance, Wider Margin. — larger C: fewer mistakes allowed

Increase Variance, Narrow margin — smaller C: more mistakes allowed

$$\Rightarrow \frac{1}{2} w^T \cdot w + C \sum_{K=1}^{n} \varepsilon_K$$

Subject to:- $y_K(w^T x_K + b) \geq 1 - \varepsilon_K, \quad \varepsilon_K \geq 0.$

- Kernel Trick: It is used so that SVM can handle non-linear data efficiently without going to high dimensions.

→ Formula for SVM optimization:

$$J(\alpha) = \sum_j \alpha_j - \frac{1}{2} \sum_i \sum_j y_i y_j \{\alpha_i \alpha_j \langle x_i, x_j \rangle$$

where :- x_i = data point i.
y_i = label of that data point (+1 or -1).
α_i = weight we solve for
→ $\langle x_i, x_j \rangle$ is dot product (similarity b/w 2 points.

$\alpha_i \geq 0 \ldots, \sum_i \alpha_i y_i = 0$,

balances the line/hyperplane correctly b/w 2 classes so the solution can't tilt unfairly to one side

→ Now Kernel Trick:

- Imagine we have a function 'φ' that maps the data into another space.
- We want to optimize the dot product of two feature vectors.
- By transforming it to φ, instead of computing $\langle x_i, x_j \rangle$, we have to compute:

$$\Rightarrow \boxed{(\varphi(x_i) \cdot \varphi(x_j))}$$

Here we apply transformation function to both data points and dot product to them

$\Rightarrow K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$
↳ Kernel Function

Now we have :-

$$\min \alpha = -\sum \alpha_i + \frac{1}{2} \sum_{i=1} \sum_{j=1} y^i y^j \alpha^i \alpha^j K(\langle x^i, x^j \rangle)$$

- **Kernel Types =**

1) **Polynomial Kernel =**

- It is a type of Kernel Function used in SVM to allow non-linear boundaries. Instead of mapping data to higher dimension explicitly, it computes the similarity b/w 2 points as if they were transformed into a higher dimension.

- $K(x_i, x_j) = (\langle x_i, x_j \rangle + b)^d$
  where - d is degree of polynomial.
  - b is a constant that controls the influence of higher-order vs lower-order terms.

2) **Gaussian Radial Basis function Kernel =**

- RBF Kernel turns the similarity b/w 2 points into an exponential function of their distance, making it possible for SVM to form very flexible, non-linear boundaries.
- Kernels in SVM are basically similarity functions that tell "how close 2 data points are in meaning or position"

- Formula for similarity Function is

$$\varphi(x^i, L) = \exp(\gamma \|x^i - L\|)$$

↓

It is feature map that maps input x^i into a new space using some landmark point L.

- Gamma HyperParameter (γ) =

It is a parameter that controls how much influence a single training point has

→ Purpose :-

1) Control similarity spread:
   - If γ is small, similarity curve will be wide. Each point influences a large area. The decision boundary becomes smooth.
   - If γ is large, curve is narrow. Decision boundary becomes tight and complex.
2) = Small γ → high bias, low variance. Model may underfit.
   - Large γ → low bias, high variance. Model may overfit.

3) Cosine Similarity Kernel = (Not important)

$$\varphi(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \|x_j\|}$$

- To measure similarity of text documents.

- SVM for Regression : (Multi-Class SVM)

- We can use SVMs for regression as well instead of classification.
  - The trick is to reverse the objective.
  - Instead of trying to fit the largest possible margin b/w 2 classes while limiting violations,
  - SVM regression tries to fit in as many instance as possible on the margin/street while limiting margin violations.

13 K-Means

- Supervised learning:

discovers the patterns in the data that relate data attributes with a target (class) attribute.

- Unsupervised learning:

The data has no target attribute as it explores the data to find some intrinsic structure in them.

- Clustering:

- It is a technique for finding similarity groups in data called clusters.
- It is often called an unsupervised learning task as no class values denoting a prior grouping of the data instances are given

- Aspects of clustering:

1) Clustering Algorithm:
   - Partitional Clustering:- Divides data into non-overlapping groups (like K-Means).
   - Hierchical Clustering:- Creates a tree of clusters, merging or splitting groups step by step.

2) • Distance (Similarity/Dissimilarity) Functions:-
   → Measures how close or far the data points are

3) Clustering Quality=

- Inter-Cluster Distances:- Should be large (clusters are far apart).
- Intra-Cluster Distances:- should be small (points with a cluster are close.

4) Quality of Clustering depend on algorithm, Distance function and the application

- K-Means Clustering=

- It is partition clustering algorithm.
- let the set of data points (or instances) D by $\{x_1, x_2, \ldots, x_n\}$ is a vector in real-valued space $X \subseteq R^r$ and 'r' is the the number of attributes (dimensions) in the data.
- The K-means algo. partitions the data (given) into 'K' cluster. Each cluster has a cluster center called clusteriod. K is specified by user.

-> Steps for how it works=

1. Decide how many groups you want the data to be devided into.
2. Randomly pick K data points from dataset as the starting 'centers' of clusters.
3. For each data point, calculate its distance from every centeriod and assign it to the closest one
4. For each cluster, calculate avg. of all data points in the cluster and set that mean (avg.) as new clusteriod.
5. Keep reassigning points and updating centeriods until centeriods stop changing significantly
6. Dataset now is divided into K-clusters.

→ K-mean pseudocode = K-means(K, D)

1) Choose k data points as the initial centriods.

2) repeat

for each data point $x \in D$ do.

compute distance from x to each centriod;

assign x to closest centriod.

endfor

recompute the centroids using current cluster memberships.

until the stopping criterion is met.

→ Stopping / Convergence Criterion =

- It is no (or minimum) reassignments of data points to different clusters.
- No (or min.) change of centriods or,
- minimum decrease in Sum of Squared error (SSE)

$$\Rightarrow \boxed{SSE = \sum_{j=1}^{K} \sum_{x \in C_j} dist(x, m_j)^2}$$

- C_j is the jth cluster, m_j is the centriod of cluster C_j (the mean vector of all data points in C_j), and $dist(x, m_j)$ is the distance b/w data points (x) and centriod m_j.

→ Formula for Centriod of Cluster (Mean) = (In Euclidean Space)

$$m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

- m_j : (mean) centriod
- $\frac{1}{|C_j|}$: Number of data points in cluster j
- $\sum_{x_i \in C_j}$: Summation of all data points in cluster
- x_i : Single data point in cluster C.

- Disk Version of K-means: It is used when dataset is too large to fit into memory.
  - Instead of loading whole data set, data is stored on disk.
  - Small chunks of data are read from disk one at a time.
  - For each batch, points are assigned to the nearest centriods and centriods are updated incrementally.
  - This process continues until convergence

- Pros of K-means:
  - Simple: easy to understand and implement.
  - Efficient: as time complexity is $O(tKn)$, where 'n' is num. of data points, 'K' is number of clusters and 't' is num. of iterations.

- Cons of K-means:
  - only applicable if mean is defined.
  - For categorical data, centriod is represented by most frequent values.
  - Sensitive to outliers.

    → We can solve this problem by either:
    1) remove those data points that are far away from centriods.
    2) Do random sampling to reduce the chances of outliers to be selected.

- Clusters of Arbitrary Shapes:

- Hyper-elliptical and hyper-spherical clusters are usually easy to represent.
- But irregular shaped clusters are hard to represent hence they may not be suitable/useful in some applications.
- Solution to this problem is Hierarchical clustering.

- Hierarchical Clustering:

- It produces a nested sequence of clusters, a tree called Dendrogram.
- Types of Hierchical Clustering:

1) Agglomerative (bottom up) Clustering:-
   - It builds tree from bottom level and merges most similar pair of clusters.
   - It stops when all data points are merged into single cluster.

2) Divisive (top down) Clustering:- (the root)
   - It starts with all data points in one cluster.
   - Splits root into a set of child clusters. Each child cluster is divided further recursively.
   - Stop when only singleton clusters of individual data points remain.

- Measuring the Distance b/w two clusters:

→ There are a few ways to do that:

① Single link:- $O(n^2)$.

- It measures the distance b/w two closest data points in 2 clusters, one data point from each.
- It can find arbitrary shaped clusters, but it may cause undesirable "chain effect by noisy points.

② Complete link:- $O(n^2 \log n)$.

- The distance is measured b/w the 2 furthest data points in the two clusters.
- It is sensitive to outliers because they are far away.

③ Average link:- $O(n^2 \log n)$.

- It is solution to Complete-link and single-link (problems) as it finds the distances b/w all datapoints
- Then it take average of all those pair-wise distances

④ Centriod Method:-

- Measures the distance b/w the centroids of 2 clusters.

- Distance Functions: (D.F)

- Key to clustering (measure how similar/dissimilar are 2 datapoints)
- There are numerous distance functions for different type of data
  1. Numeric data
  2. Nominal data
  3. Different specific application.

- D.F for Numeric attributes:

- Most commonly used ones are:
  - Euclidean
  - Manhattan.
- Denoted by $dist(x_i, x_j)$ → x_i, x_j are datapoints
- Minkowski Distance is generalized distance metric that includes above two ones are well. It is defined as:-

$$D(x,y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{1/p}$$

where;
- → x, y are i^{th} data points.
- → p is order of norm (a parameter.
  - → $p = 1$ → Manhattan distance
  - → $p = 2$ → Euclidean distance.
  - → $p = \infty$ → Chebyshev distance
- when $p = 1$:-

⇒ $dist(x_i, x_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| \cdots + |x_{ir} - x_{jd}|$

- when $p = 2$:-

$\Rightarrow dist(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 \cdots + (x_{ir} - x_{jr})^2}$

- Weighted euclidean distance :-

$\Rightarrow dist(x_i, x_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 \cdots + w_r(x_{ir} - x_{jr})^2}$

- Squared Euclidean Distance is another type where the under root is removed from normal Euclidean Dist. It is used cause it requires less computation than normal one and gives similar results.

- Chebyshev Distance :-

$dist(x_i, x_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, |x_{ir} - x_{jr}|)$

Chooses maximum distance among all the distances calculated.

- Distance Function for Binary and Nominal Attributes :-

- Binary attributes : has two values or states but no ordering relationship, eg: Gender (Male or Female).
- We use confusion matrix to introduce the distance functions/measures.

- Confusion Matrix :- Compares 2 data points based on their binary attributes (0 or 1) to measure similarity or dissimilarity.

Data point (i) \ Data point (j)	1	0
1	a	b
0	c	d

- a = no. of attributes where both i and j are 1
- b = no. of attributes where i = 1, j = 0
- c = no. of attributes where i = 0, j = 1
- d = no. of attributes where both i and j are 0.

- Once matrix is constructed, we calculate the similarity / disimilarity b/w 2 data points.
- We use a, b, c, d from the confusion matrix to calculate Distance functions.
- Distance functions for this are:-

(1) → Simple matching Coefficient :-

$$SMC = \frac{b+c}{a+b+c+d}$$

→ It is for symmetric Binary attributes where (0 and 1) states have equal importance and carry the same weight

(2) → Jaccard Coefficient :-

$$JC = \frac{b+c}{a+b+c}$$

→ - It is for Assymetric Binary attributes where one of the states is more important than the other.
- By convention state 1 is more important state which is typically rare or infrequent state

- Nominal Attributes with more than two states.

$$dist(x_i, x_j) = \frac{r-q}{r}$$

where → r is total number of attributes
q is number of values that match in both x_i and x_j.

- Data Standardization:

- In the euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- It forces attributes to have a common value range.

- Interval-scaled Attributes:

- for numerical attributes
- Their values are real numbers following a linear scale.
- Eg Age of 10 and 20 have equal difference as 40-50 have.
- Two main approaches to standardize interval scaled attributes are.

① **Range:-**

$$\Rightarrow \boxed{range(x_{if}) = \frac{x_{if} - min(f)}{max(f) - min(f)}}$$

where
- x_{if} = value of feature f for instance i.
- max(f) = max. value of feature f in dataset.
- min(f) = min. " " " " " "

② **Z-score:-**

Transforms the attributes values so that they have a mean of zero and a mean absolute deviation of 1. The mean absolute deviation of feature f, denoted by s_f, is computed as follows:

$$\Rightarrow \boxed{s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)}$$

where $\Rightarrow m_f = \frac{1}{n}(x_{1f} + x_{2f} + \dots + x_{nf})$.

$\Rightarrow$ hence:- $$\text{Z-score} = \frac{x_{if} - m_f}{s_f}$$

- Ratio-Scaled Attributes:
  - Numeric attributes, but unlike interval-scaled attributes, their scale are exponential.
  - For example: Total amount of microorganisms that evolve in a time t is approximately given by Ae^{Bt}, where A and B are constants.
  - Solution is log Transform: $(\log[x_{if}])$.
    $\Rightarrow \log(Ae^{Bt}) = \log(A) + Bt$.
  - After log transformation, treat result as linear (interval-scaled attribute) and apply range or Z-score.

- Nominal Attributes:
  - We need to transform nominal attributes to numeric attributes.
  - Transform nominal to binary attributes.
    1) The number of values of a nominal attribute is V.
    2) Create V Binary attributes to represent them.
    3) If a data instance for a nominal attribute takes a particular value, the value of its binary attribute is set to '1', otherwise '0'.
    $\Rightarrow$ like if we have Red, Blue, Green.
    if a Row (data instance) is Red then if it take Red then the value will be '1'; else if it is Green or blue in it, it will be '0' in these 2 cases.

- Ordinal Attributes:
  - It is like a nominal attribute, but its values have a numerical order like:
  - Young, Middle age, and Old in Age attribute. They are ordered.
  - We treat them as interval-scaled attributes.

- Mixed Attributes:
  - Our distance functions are either for all numeric or all nominal, etc.
  - Practical data has different types:
    - Any subset of following.
    ① interval - scaled ② symmetric
    ③ Assymetric ④ ratio-scaled
    ⑤ Ordinal ⑥ Nominal.
  - How to deal with it?:-

→ Convert mixed data types to One type:
  - Pick most common datatype and convert others to match it.
  - Converting to numerical:-
    - easy ones are=
      ① Ordinal → Numerical.
      ② Ratio-scale → log transform → Numerical
      ③ Binary → 0 or 1.
    - Problematic =
      - Nominal (colors) → Arbitrary numbers (Red=1, Blue=2)
      - Problem is we are creating fake ordering where it doesn't exist. Instead we can convert it to symmetric binary, which are then treated as numeric.

- Combining Individual Distances:

$$\Rightarrow \boxed{dist(x_i, x_j) = \frac{\sum_{f=1}^{r} \delta_{ij}^{f} \, d_{ij}^{f}}{\sum_{f=1}^{r} \delta_{ij}^{f}}}$$

where f = attribute
d_{ij}^{f} = distance for attribute f b/w i and j.
δ_{ij}^{f} = weight for attribute f (indicates importance).

⇒ Allows different attributes to have importance levels and handles mixed data types by computing appropriate distance measures for each attribute type separately

- Cluster Evaluation:

- The quality of clustering is a hard problem as it is hard evaluate because we donot know the correct clusters.
- Some methods used are:
  1. Study centriods, and spreads.
  2. Rules from a decision tree.
  3. For text docs, one can read some docs in the clusters.
- Another way is Ground Truth:-
  → We use some labeled data (for classification)
  → Assumption: Each class is a cluster.
  → After clustering, a confusion matrix is constructed.
  → From matrix, we compute entropy, purity, precision, recall and F-score.

- let classes in data 'D' be $C = \{c_1, c_2, c_3 \cdots, c_k\}$
- The clustering method produces K clusters, which divides 'D' into 'K' disjoints subsets $D_1, D_2 \cdots D_k$
- Entropy for each cluster is measured by:

$$\text{Entropy}(D_i) = -\sum_{j=1}^{K} Pr_i(c_j) \log_2(Pr_i(c_j))$$

(tells uncertainty in dataset)

where • $Pr_i(c_j)$ is probability of outcome c_j occuring.
• c_j is class in dataset.

- Purity measures the extent that a cluster contains only one class of data. It basically tell proportion of most frequent class in that cluster.

$$\text{Purity}(D_i) = \max_j (Pr_i(c_j))$$

where, D_i = specific cluster.
max = find largest value.

14 Principal Component Analysis (PCA)

- PCA is a dimensionality reduction technique used in ML and statistics. The main idea is:
- Data often has many features (dimensions), but some of them may be redundant or less important.
- PCA transforms the data, into a set of features called principal components which are combinations of the original features.
- These components are ordered by how much variance (information) they capture.
- Purpose of PCA:
  1. Reduce dimensionality.
  2. Remove noise/redundancy in data.
  3. Improve visualization by converting higher dimensions in 2D or 3D.
  4. Speed up algorithms by working with fewer features.

- PCA Drawbacks:

1) PCA assumes that the variables are linearly correlated. If the correlations is not linear, then PCA will not be efficient.
2) PCA performs lossy compression, which means that the information is lost when we discard insignificant components.
3) Since each principal component is a linear combination of original features, visualizations are not easy to interpret or relate to original features.

④ PCA results depend on the scale of data. If features are measured in different unit or have high ranges, PCA may give more weight to the larger-scaled feature. Therefore, data should be standardized before applying PCA.

- Algorithm Steps:-

→ Suppose $x_1, x_2, \cdots, x_M$ are N-Dimensional vectors
→ So x is an $N * M$ matrix.
→ N is no. of Dimensions and M is number of samples.

① Compute Data mean:-

$$\Rightarrow \boxed{\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i}$$

② Subract mean from each row of X (centering data):-

$$\Rightarrow \boxed{\phi_i = x_i - \bar{x}}$$

③ Form the matrix A:-

$$\Rightarrow A = [\phi_1, \phi_2, \cdots, \phi_M] \ (N * M \text{ matrix})$$

④ Compute the covariance matrix C:-

$$\Rightarrow \boxed{C = \frac{1}{M} \sum_{n=1}^{M} \phi_n \phi_n^T = AA^T \ (N * N \text{ matrix})}$$

⑤ Compute the Eigen Values of C:-

$$\lambda_1 > \lambda_2 > \lambda_3 > \cdots \lambda_N$$

$$\Rightarrow \text{Eigen Values} \Rightarrow \boxed{\det(C - \lambda I) = 0}$$

⑥ Compute eigen Vector of C using those Values:-

$$u_1, u_2, \cdots, u_N.$$

$$\Rightarrow \boxed{\text{Eigen Vectors} = (C - \lambda I)(u) = 0.}$$

(u → EV-Matrix)

- Since C is symmetric, $u_1, u_2, \cdots, u_N$ form a basis

- Any vector $\rightarrow [\phi_i = x_i - \bar{x}]$ can be rewritten as a linear combination of the eigen vectors:-

$$\Rightarrow \boxed{\phi_i = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i}$$

$$\Rightarrow \therefore \boxed{b = u^T \phi} \quad \because b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

- b contains the coefficients (weights) that tell you how much of each eigenvector is present in ϕ_i.
- So basically Principal Components = Eigen Vectors.

(7) Dimensionality Reduction Step:- Keep only the terms corresponding to the K largest eigen values where $K < N$

$$\Rightarrow \boxed{\sum_{i=0}^{K} b_i u_i}$$

Note ⇒ When we are done with computation of eigen values, arrange them in descending order and reorder eigen vectors accordingly as it is important for step 7, because we need most important components in the beginning.

- PCA SVD =

- In PCA Singular Value Decomposition, we directly decompose data matrix X (after mean-centering)
- Calculating covariance matrix by multiplying matrix by its transpose can cause loss of precision and numerical errors.
- Hence we use SVD technique:

$$\Rightarrow \boxed{X = USV^T}$$

where;

- U = left singular vectors (related to data in sample space. $(n \times n)$
- S = Diagonal matrix of singular values (magnitude of variance in each direction). $(n \times p)$
- V = Right singular vectors, which are the principal components (directions of maximum variance. $p \times p$

- In short → 'V' gives directions of PCs.
  → 'S' gives importance of PCs.
  → 'U' gives coordinates of data in new PC space

- Main Applications of PCA =

1) Image Compression.
2) Preprocessing step for classification/regression/clustering tasks.
3) Facial Recognition: Eigen Faces
4) Data Visualization