ML

PAGE No.
DATE / /

1. Predict the price of the Uber ride from a given pickup point to the agreed drop off location.
Perform following tasks:
(i) Pre-process the dataset
(ii) Identify outliers.
(iii) Check the correlation
(iv) Implement linear regression and random forest regression model.
(v) Evaluate the models and compare their respective scores like R2, RMSE, etc.

Objectives: (1) Understand the dataset & cleanup
(2) Build Regression models to predict the fare price of uber ride.
(3) Also evaluate the models & compare their respective scores like R2, RMSE, etc.

→ Data Preprocessing Steps:
① Get the dataset
② Import the libraries
③ Import the dataset
④ Check out missing values
⑤ See the Categorical Values
⑥ Splitting the dataset into Training & Test Set
⑦ Feature Scaling [Put variables in same range & scale so that no variable dominate other]
[Standardization, Normalization]
mean=0, s.d.=1 [0, 1] or [-1, +1]

$$\frac{x-u}{\sigma}$$

$$\frac{x-\min}{\max-\min}$$

less sensitive, highly sensitive to outliers
No upper lower bound fixed

KNN, neural nets

Simple Linear Regression: If single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

Multiple Linear Regression: If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Cost Function: Minimize error. It provides best possible values for b_0 & b_1 to make best fit line for the data points.

$$Y = b_0 + b_1 x + e$$

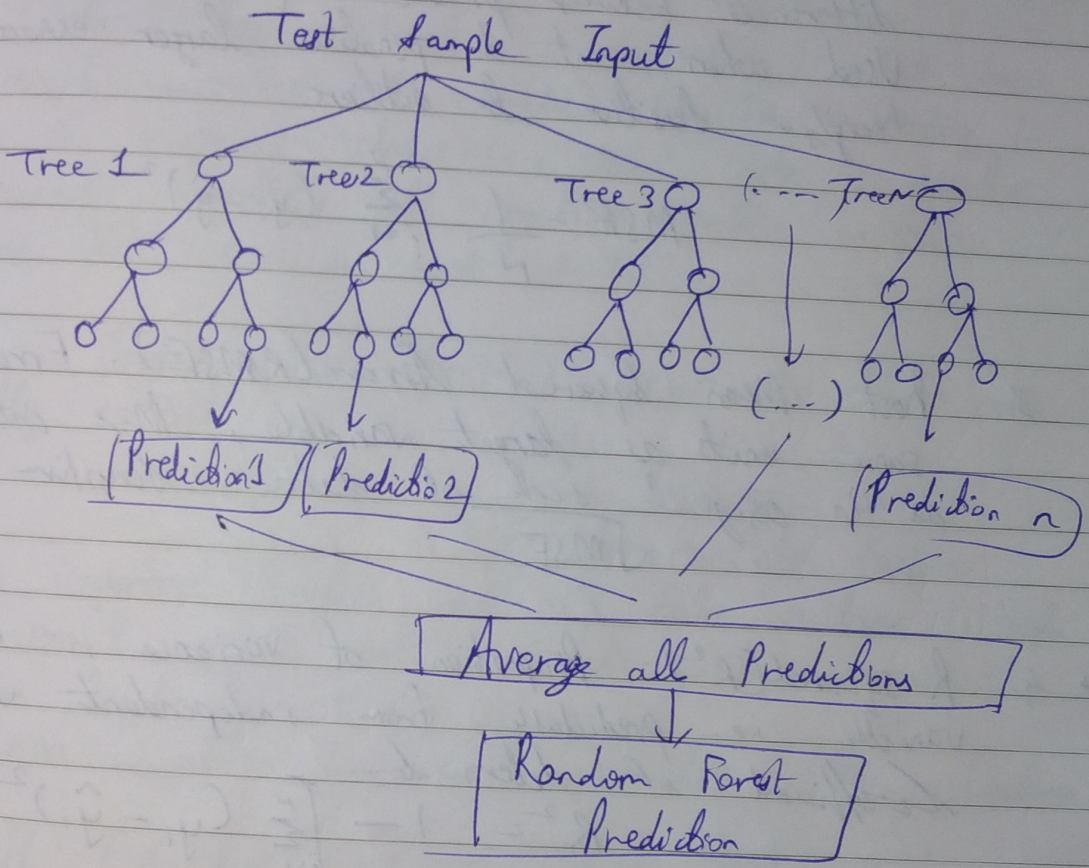
- Y: dependent variable
- b_0: Y intercept
- b_1: slope
- x: independent variable
- e: error

Gradient Descent: Method of updating b_0 & b_1 to reduce MSE. Idea is to keep iterating the b_0 & b_1 values until we reduce MSE to minimum.

Numerical → Numbers, arithmetic operation allowed, continuous or discrete, e.g. age, income.

categorical → Categories or labels, nominal or ordinal, e.g. gender, marriage status

Random Forest: Supervised learning Algorithm that uses ensemble learning method for classification & regression. It is bagging technique not a boosting technique. Trees in random forests run in parallel.



Random Forest creates multiple subsets of original data by randomly sampling with replacement.

Advantages: High Accuracy, feature importance, suitable for both classification & regression, can handle both numerical & categorical data,

Evaluation scores for Regression Models:

1) Mean Absolute Error (MAE) — Avg. of absolute differences btwn predicted & actual values. More interpretable.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

2. Mean Squared Error (MSE) — Avg. of squared differences between predicted & actual values. Used when want to penalize larger errors more heavily. Sensitive to outliers.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y})^2$$

3. Root Mean Squared Error (RMSE): Error in same units as target variable. More interpretable as it is in original scale of data. Sensitive to outliers

$$\sqrt{MSE}$$

4. R-squared (R^2): Proportion of variance in dependent variable i.e. predictable from independent variables. Coefficient of determination

$$R^2 = 1 - \left[\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \right]$$

Useful to determine how well model explains variance in target variable. Commonly used in linear regression. Misleading when comparing models with different no. of predictors. Model can have high R^2 but still perform poorly if it's overfitting.

Evaluation Scores for Classification Models

1. Accuracy: Proportion of correct predictions (both true positives & true negatives) out of all predictions.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Predictions}}$$

Useful when class distribution is balanced (i.e. both classes have roughly same no. of examples). Can be misleading in cases of imbalanced classes (e.g. 95% data is negative, so model predicting all negatives would have 95% accuracy but it's a bad model)

2. Precision: Proportion of positive predictions that were actually correct.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

It is important when cost of false positive is high (e.g. email spam where false positive means marking legitimate email spam)
Precision alone doesn't consider false negatives.

3. Recall (Sensitivity or True Positive Rate): Proportion of actual positives that were correctly identified by model.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Important when cost of false negative is high (Medical diagnosis, where failing to detect disease could be critical).
High recall can lead to many false positives.

4. F1- Score : Harmonic mean of Precision & Recall, balancing 2 metrics.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Useful when you need a balance between precision & recall and where there's an imbalanced class distribution.
Does not provide direct interpretation of individual performance of precision or recall.

5. ROC-AUC [Receiver Operating Characteristic — Area Under Curve] — AUC represents area under ROC curve which is plot of true positive rate (recall) vs false positive rate across different thresholds. AUC is useful when you need to evaluate the model across different classification thresholds. It provides a single value that summarizes the model's performance. It can be misleading with highly imbalanced datasets.

6. Confusion Matrix: Detailed Breakdown of model's performance & help identify specific problems in model (like more F.P. or F.N.).

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

PAGE No.
DATE / /

2. Email Spam Detection

Classify the email using the binary classification method. Email Spam detection has two states:
(a) Normal State – Not Spam (b) Abnormal State – Spam.
Use K-Nearest Neighbors & Support Vector Machines for classification. Analyze their performance.

Objectives: Understand the Dataset & cleanup (if required)
Build K-Nearest Neighbour & SVM models to predict the spam email.
Also evaluate the models & compare their respective scores like R^2, RMSE, etc.

- K-Nearest Neighbors (KNN) – Supervised ML Algorithm used for classification, regression, outlies detection.
Each time there is new point added to data, KNN uses just one part of data for deciding value (regression) or class (classification) of that added point.
Since it doesn't have to look at all the points again, this makes it a lazy learning algorithm.
KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about underlying data.
This is an extremely useful feature since most of the real-world data doesn't really follow any theoretical assumption.
E.g. Uniform Distribution, Linear separability, etc.
Working:
The algorithm first calculates the distance of a new data point to all other training data points.
After calculating the distance, KNN selects a number of nearest data points – 2, 3, 10 or any integer.
This number of points (2, 3, 10, etc) is the K in

K-Nearest Neighbors

In final step, if it is a regression task, KNN will calculate the average weighted sum of the K-nearest points for the prediction.
If it is classification task, the new data point will be assigned to the class to which the majority of the selected K-nearest points belong.

- KNN Classification:

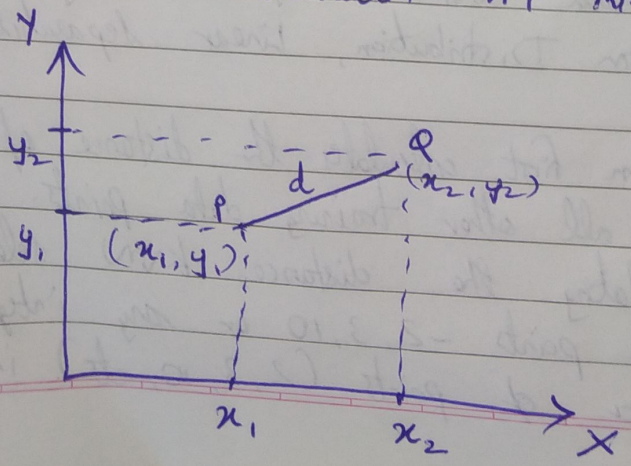
→ Step 1: Collect Data (Labelled)
Step 2: Preprocess Data.

E.g. Predicting Movie Genre

IMDb Rating	Duration	Genre
8.0 (Mission Impossible)	160	Action
6.2 (Gadar 2)	170	Action
7.2 (Rocky & Rani)	168	Comedy
8.2 (OMG2)	155	Comedy

Data collect Labelled →

Predict the genre of "Barbie" movie with IMDb rating 7.4 and duration 114 minute.



Step 3: Calculate Distances.

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Calculate the Euclidean distance between the new movie & each movie in the dataset.
Barbie (7.4, 114)

Distance to (8.0, 160) = $\sqrt{((7.4 - 8.0)^2 + (114 - 160)^2)} \approx 46.00$
Distance to (6.2, 170) = $\sqrt{(7.4 - 6.2)^2 + (114 - 170)^2} \approx 56.01$
Distance to (7.2, 168) = $\sqrt{(7.4 - 7.2)^2 + (114 - 168)^2} \approx 54.00$
Distance to (8.2, 155) = $\sqrt{(7.4 - 8.2)^2 + (114 - 155)^2} \approx 41.00$

Step 4: Select K Nearest Neighbours

Let K = 3 here.
∴ Nearest set here will be (8.2, 155), (8.0, 160), (7.2, 168)

Step 5: Majority Voting (Classification)

Comedy, Action, Comedy
∴ Majority is Comedy.

∴ Barbie will be classified as Comedy.

Step 6: Test accuracy (confusion matrix) of result & visualize test set result.

KNN Regression: Continuous data & numeric.

Same steps.

E.g.

Data Point	Feature X	Target Y
1	2.0	5.0
2	4.0	8.0
3	6.0	12.0
4	8.0	15.0
5	10.0	20.0

Now consider new data point with $x_1 = 7.0$
Using KNN with $K=2$, predict the target value (Y) for this new data point.

Distance to 2.0 $= \sqrt{(7-2)^2} = 5$

4.0 $= \sqrt{(7-4)^2} = 3$

6.0 $= \sqrt{(7-6)^2} = 1$

8.0 $= \sqrt{(8-7)^2} = 1$

10.0 $= \sqrt{(7-10)^2} = 3$.

At $K=2$, selecting data point 3, 4,
averaging target Y for both

$$\frac{12+15}{2} = \underline{\underline{13.5}}$$

∴ Target value Y for $x_1 = 7.0$ will be 13.5

Support Vector Machine: Supervised learning algorithm used for classification as well as regression. Goal is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in correct category.
This best decision boundary is called hyperplane.
SVM chooses extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors & hence algorithm is termed as Support Vector Machine.
SVM can be of 2 types:
① Linear SVM: Used for linearly separable data, which means if a dataset can be classified into 2 classes by using single straight line, then such data is termed as linearly separable data & classifier is used called as Linear SVM classifier.
② Non-Linear SVM: used for non linearly separated data, which means if dataset cannot be classified using straight line, such data is termed as non-linear data & classifier is called non linear SVM classifier.

Hyperplane — There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out best decision boundary that helps to classify the data points. This best boundary is known as hyperplane of SVM.
The dimensions of the hyperplane depend on the features present in the dataset, which means

if there are 2 features, then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be 2 dimensional plane.
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors: The data points or vectors that are the closest to the hyperplane & which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a support vector.
Support vectors are data points that are closer to the hyperplane & influence the position & orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane.
These are the points that help us build our SVM.
In SVM, we take the output of the linear function & if that output is greater than 1, we identify it with one class and if output is -1, we identify is with another class.
Since the threshold values are changed to 1 & -1 in SVM, we obtain this reinforcement range of values ([-1, 1]) which acts as a margin.

Cost Function & Gradient Updates: In the SVM algorithm, we are looking to maximize the margin between the data points & the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(w)) = \begin{cases} 0, & \text{if } y * f(w) \geq 1 \\ 1 - y * f(w) & \text{else} \end{cases}$$

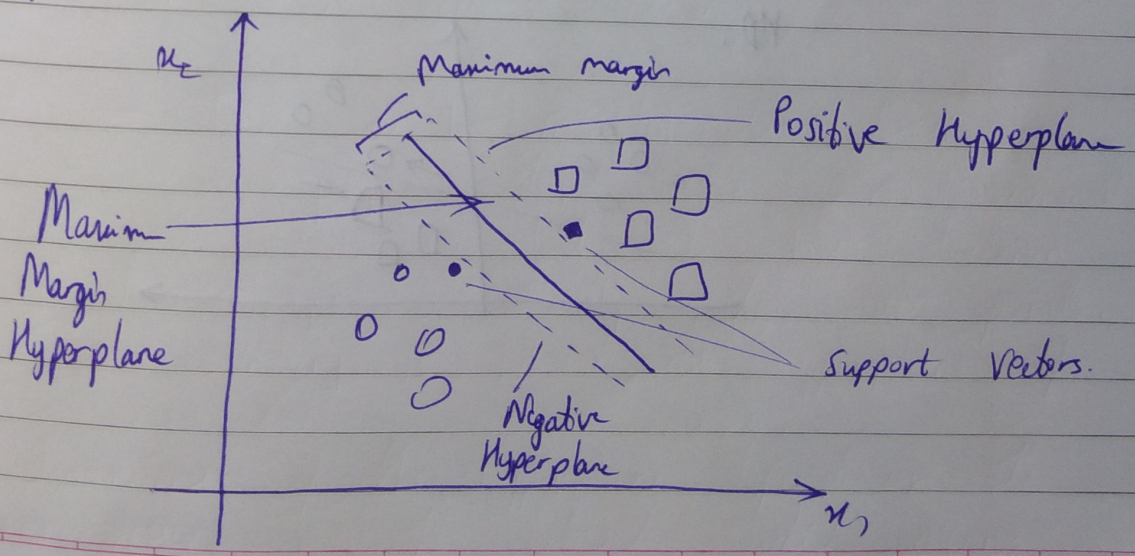
Hinge loss function.

The cost is 0 if the predicted value & the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below.

$$\min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i (w \cdot x_i + b))$$

Gradient Update → No misclassification (model predict correct class)

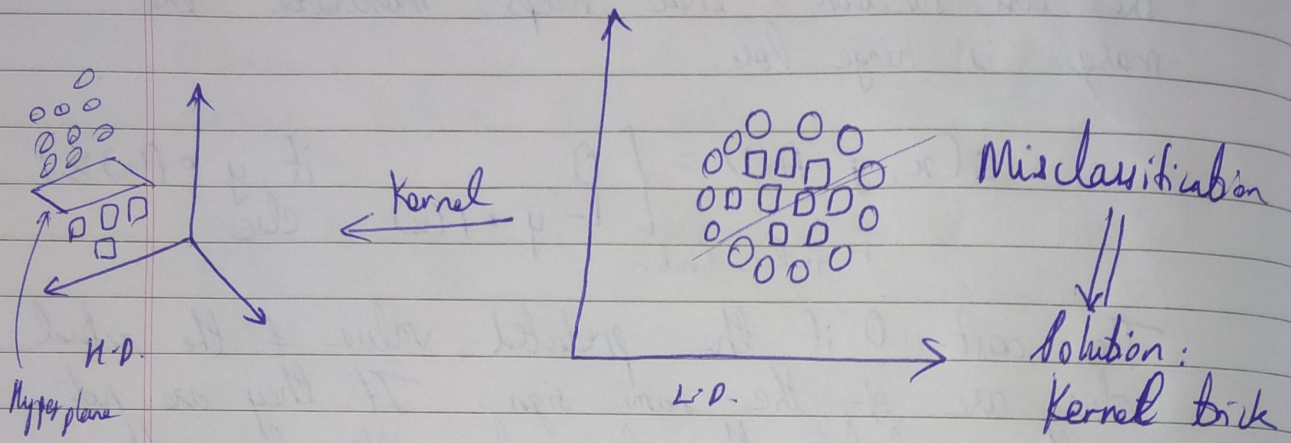
Gradient



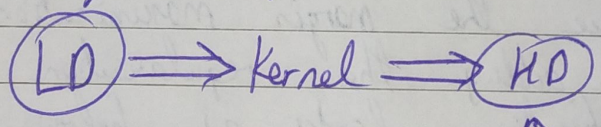
Kernel functions, Gradient Updates, Cost function.

Non linear SVM: ~~Non~~ Data can't be separated linearly / using straight line

E.g.

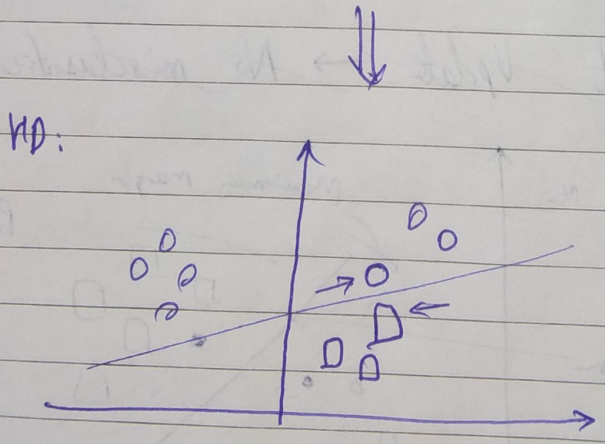


Kernel function: Takes the low dimensional feature space & gives high dimensional feature space as output.



data becomes separable

E.g. LD: ○○○○ □□□ ○○○



Sepolia Faucet Google cloud.

PAGE No.
DATE / /

Kernel functions, Gradient Updates, Hinge Loss Cost function.

PAGE No.
DATE / /

- Why not to use KNN?
  (i) Computational Cost
  (ii) Memory Intensive
  (iii) Sensitivity to Noise
  (iv) ~~large~~ Poor performance for large datasets (high dimensionals)

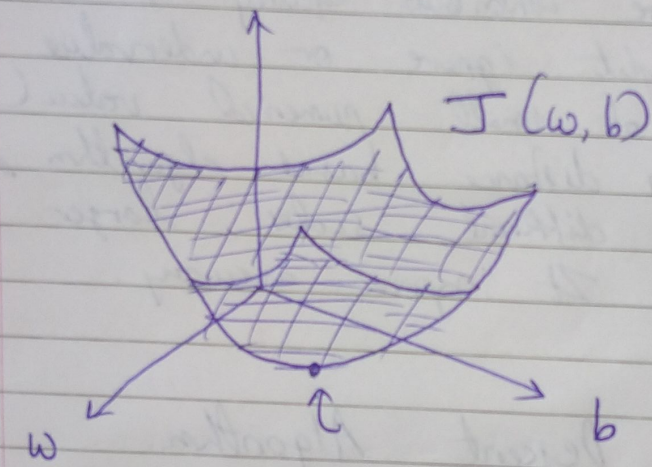
- Feature scaling required in KNN because it ensures all feature contribute equally to distance matrix. It might ignore or undervalue important features because of small numerical value (if not scaled). It is distance based algorithm so if features have different scales, larger scales can dominate. It improve accuracy.

3 - Gradient Descent Algorithm.

→ It is optimization algorithm for finding a local minimum of a differentiable function.
→ Gradient descent in ML is used to find values of function's parameters (coefficients) that minimize a cost function as far as possible.
→ Gradient descent is optimization algorithm that's used when training a machine learning model.
→ It's based on convex function & tweaks its parameters iteratively to minimize a given function to its local minimum.
→ In ML, a gradient is derivative function that has more than one input variable.
→ Gradient, also known as slope of function, is simply

measures the change in all weights with regard to change in error.

- Imagine you have a machine learning problem & want to train your algorithm with gradient descent to minimize your cost-function $J(w,b)$ & reach its local minimum by tweaking its parameters (w and b).
Gradient descent is convex function



We want to find values of w & b that correspond to minimum of cost function.
To start finding the right values we initialize w & b with some random numbers.
Gradient descent then starts at that point (somewhere around top of illustrn.) & it takes one step after another in the steepest downside direction (i.e. from the top to the bottom of the illustrn.) until it reaches the point where the cost function is as small as possible.

Convex function → If you take any 2 points on graph of function & draw a straight line btwn them, line will always be above or on graph of function.

Function Requirements: Gradient descent algorithm does not work for all functions. There are 2 specific requirements. A function has to be:
(i) Differentiable
(ii) Convex.

Algorithm — Iteratively calculates the next point using gradient at the current position, scales it (by a learning rate) and subtracts obtained value from the current position (makes a step). It subtracts the value because we want to minimize the function (to maximize it would be adding). This process can be written as:

$$p_{n+1} = p_n - \eta \nabla f(p_n)$$

η scales the gradient & thus control the step size. In ML, it is called learning rate. & have strong influence on performance.

The smaller learning rate, longer the GD converges, or may reach maximum iteration before reaching the optimum point.
If learning rate is too big, algorithm may not converge to the optimal point (jump around) or even to diverge completely.

Gradient Descent Method's steps are:

1) Choose a starting point (initialisation)
2) Calculate gradient at this point
3) Make a scaled step in the opposite direction to the gradient (objective: minimise)
4) Repeat points 2 & 3 until one of the criteria is met:
   (i) Maximum number of iterations reached
   (ii) Step size is smaller than tolerance (due to scaling or small gradient).

This function takes 5 parameters:
(i) Starting Point – In our case, we define it manually but, in practice, it is often a random initialisation.
(ii) Gradient Function – Has to be specified beforehand.
(iii) Learning Rate – Scaling Factor for step sizes
(iv) Maximum number of iterations
(v) Tolerance to conditionally stop the algorithm.

→ Gradient Descent is method used to minimize a function. Basic idea is to start at random point on function & then move in direction of steepest decrease in function, gradually getting closer to the minimum (the lowest point) of the function.

PAGE No.
DATE / /

E.g. $f(x) = x^2$

The graph of this function is a parabola that opens upwards, & minimum value is at $x=0$, where the curve touches the x-axis

Step 1:- Start with a random point - Suppose we start at $x=3$. We want to minimize the function $f(x) = x^2$.

Step 2:- Calculate the gradient:- The gradient (or derivative) of $f(x)=x^2$ is -

$$f'(x) = 2x$$

The gradient tells us the slope of the function at any point. It shows us how much the function is increasing or decreasing.

Step 3:- Update the point: Gradient descent updates the point by moving in the direction of the negative gradient (downhill). The update rule is:

$$x_{new} = x_{old} - \text{learning rate} \times f'(x_{old})$$

The learning rate is a small number that controls how big each step is. Let's use a learning rate of 0.1 for this example.

Step 4:- First step: Start at $x=3$

Compute the gradient at $x=3$: $f'(3) = 2 \times 3 = 6$

Update the point - $x_{new} = 3 - 0.1 \times 6 = 2.4$

So, the new point is $x = 2.4$

Step 5:- Second Step: Now, we repeat the process starting at $x = 2.4$.

Compute the gradient at $x = 2.4$:

$$f'(2.4) = 2 \times 2.4 = 4.8$$

Update the point:

$$x_{new} = 2.4 - 0.1 \times 4.8 = 1.92$$

So, new point is $x = 1.92$

Step 6:- Repeat: We keep repeating these steps. With each iteration, the value of x moves closer to 0, the minimum of the function.

Why it works: At each step, gradient descent moves the point in the direction where the function decreases the most (i.e. downhill). The steps get smaller as we approach the minimum, so the algorithm converges to the minimum point over time.

	Batch Gradient	Stochastic Gradient	Mini Batch Gradient
Update Freq.	Once per full dataset (after each epoch)	Once per training example	Once per mini-batch (subset of dataset)
Computation per step	Uses all data	Use 1 data point	Uses subset of data
Convergence Speed	Smooth, but slow	Noisy, fluctuates on	Smooth & faster
	Slow for large datasets	Fast for large datas	Faster than Batch, stable than SGD
Memory	High	Low	Moderate
For large data	Not ideal	Ideal	Ideal (efficient)
Use	Small dataset, prob	Online learning	Comon in DL
Batch size	Entire dataset	1	32, 64, 128, 256

Does gradient descent coverge to an optimum.

PAGE No.
DATE / /

- How gradient descent work in linear regression?

KNN advantages:

① Simple to implement
② Robust to noisy training data.
③ Effective if training data is large.

Disadvantages:

1) Always needs to determine value of K which may be complex some time.
2) Computation cost is high because of calculating the distance btwn data points for all training samples.

Small k → low bias, high variance (overfitting)
high k → high , low (underfitting)

~~K Means Clustering.~~

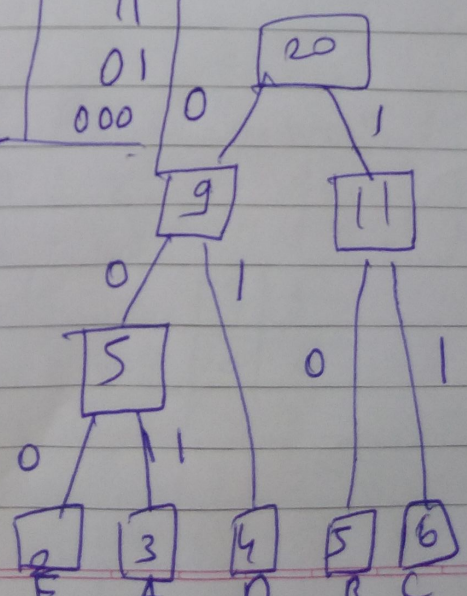
ASCII → Huffman coding.

5× 8 bit = 40 bits + 12 bits = 40+12 = 52 bits

Character	count	code
A	3	001
B	5	10
C	6	11
D	4	01
E	2	000
	20	

3×3 = 9
5×2 = 10
6×2 = 12
2×3 = 6
4×2 = 8

9+10+12+6+8
= 45 bits
↑
Message size



table/tree size
= 52 bits

∴ total
= 97

(i) Sort & arrange.
(ii) Merge 2 smallest & make 1

PAGE No.
DATE / /

$\Sigma d_i * f_i = 3 \times 2 + 3 \times 3 + 2 \times 4 + 2 \times 5 + 2 \times 6 = 45$ bit.

Huffman Encoding: Lossless data compression algorithm.
Used for reducing size of data / message.

→ Fractional Knapsack:

		✓	✓	✓	✓	✓	✓	✓
Q.	Object:	1	2	3	4	5	6	7
	Profit:	5	10	15	7	8	9	4
	Weight:	1	3	5	4	1	3	2
	P/w	5	3.33	3	1.75	8	3	2

W = 15
n = 7.

→ 1) Select item with ~~max profit / min. weight~~ / max ratio p/w

Objects	Profit	Weight	Remaining Weight
5	8	1	14
1	5	1	14 − 1 = 13
2	10	3	10
3	15	5	5
6	9	3	2
7	4	2	0
	51		

1
non profit

→ ~~0/1~~ Knapsack. [Dynamic Programming] → subproblems

Eg. weight = {3, 4, 6, 5}
(value) Profits = {2, 3, 1, 4}

W = 8
n = 4

i ↓ / w →	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	2	2	2	2	2
2	0	0	0	0	3	3	3	3	3
3	0	0	0	0	0	4	5	4	4
4	0	0	0	0	0	0	1	1	1

P_i	w_i
2	3
3	4
4	5
1	6

$x_i = 0/1$

PAGE No.
DATE / /

0/1 Knapsack Problem

sum of profits ↓

$\max \sum p_i x_i$

$\sum w_i x_i \leq m$

↑ sum of weights

$m = 8$ $\quad P = \{1, 2, 5, 6\}$

$n = 4$ $\quad w = \{2, 3, 4, 5\}$

D.P. → Sequence of Decisions to solve problem. Write all possible solutions & pick one. (2^n ← objects here)

P_i	W_i
1	2
2	3
5	4
6	5

w →

i or v ↓	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1
2	0	0	1	2	2	3	3	3	3
3	0	0	1	2	5	5	6	7	7
4	0	0	1	2	5	6	6	7	(8)

consider both

1st row → consider 1st object ignore other

2nd row → Ignore remaining consider 1st & second

⋮

i^{th} row → Consider all objects in prev. rows.

Formula → $V[i, w] = \max [V[i-1, w], V[i-1, w - w[i]] + P[i]]$

$V[4, 1] = \max \{V[3, 1], V[3, 1-5] + 6\}$

~~Consider $P_i = x_i$~~

x_1	x_2	x_3	x_4
0	1	0	1

as we got 8 only in 4th row

Remaining profit = $8 - 6 = 2$

PAGE No.
DATE / /

Quick Sort

E.g.

	0	1	2	3	4	5	6	7	8	9 (h)
A	10	16	8	12	15	6	3	9	5	∞
	i									j

Let pivot = 10

∴ we will have to find sorted position of 10. We will use i, j for this. i will search for elements which are > pivot & j will search for elements < pivot, so that they can exchange numbers.

∴ Increment i until we find element > pivot, decrement j until we find element < pivot

① i = 1, j = 8. Here, j < pivot.

∴ swap

10	5	8	12	15	6	3	9	16	∞
	1						1		

② i = 3 has element > pivot.
j = 7 " " < pivot

∴ swap

0	1	2	3	4	5	6	7	8	9
10	5	8	9	15	6	3	12	16	∞

③ i = 4 ele > pivot
j = 6 ele < pivot
swap

10	5	8	9	3	6	15	12	16	∞

i = 6 Now do not interchange as i
j = 5 has crossed j.

Position of pivot will be of j currently.

∴

6	5	8	9	3	10	15	12	16	∞

Now all elements before 10 are less, after are greater

Now break list & perform quick sort recursively.

Deterministic → Pivot chosen in fixed way. [1st, last, or middle]

Performance can be predictable, but in worst case (when list is sorted), the algorithm can take much longer time – $O(n^2)$

Randomized → Pivot chosen randomly from list, which reduces chances of picking bad pivot (like smallest or largest element in sorted list)
This randomness helps avoid worst case performance, so on average, algorithm runs much faster $O(n \log n)$, even in cases where list might be nearly sorted.

K-means: Unsupervised, aims to partition n observations into k clusters in which each observation belongs to cluster with nearest centroid. The algorithm aims to minimize squared Euclidean distances between the observation & the centroid of cluster to which it belongs. K-means clustering is an unsupervised learning algorithm which aims to partition n observations into k clusters in which each observation belongs to cluster with nearest centroid. The algorithm aims to minimize the squared Euclidean distances between the observation & the centroid of cluster to which it belongs.

Advantages: ① Efficient. Time complexity $O(nkt)$ where n = no of instances, k = no of clusters, t = no of iterations
② Often terminates at local optimum

Disadvantages – ① Requires to specify no. of clusters (k) in advance
Can't handle noisy data & outliers. Not suitable to identify clusters with non-convex shapes

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

Eg.

Sr_no	age	amount
C_1	20	500
C_2	40	1000
C_3	30	800
C_4	18	300
C_5	28	1200
C_6	35	1400
C_7	45	1900

$k = 2$

Let $k_1 = (20, 500)$ $k_2 = (40, 1000)$

$d(C_3, k_1) = \sqrt{(30-20)^2 + (800-500)^2} \approx 300.$

$d(C_3, k_2) = \sqrt{(30-40)^2 + (800-1000)^2} \approx 200.$

$\therefore C_3$ will go into k_2.

Now new centroids: $k_2' = \left(\frac{40+30}{2}, \frac{800+1000}{2}\right)$

$= (35, 900).$

$d(C_4, k_1) =$

$d(C_4, k_2) =$

PAGE No.
DATE / /

Hierarchical clustering: Group similar things together into clusters. Method — (i) Start with each object in its own group (ii) Gradually merge closest group based on similarity (iii) Repeat process until all objects in one big group. 2 types: Bottom up, top down
It is slower, robust to outliers

Elbow Method —