DSV
unit-4
Advanced Data Analysis.

Decision Tree !

- It is a supervised machine learning algorithm that is used for both classification & regression tasks. It models decisions and their consequences.

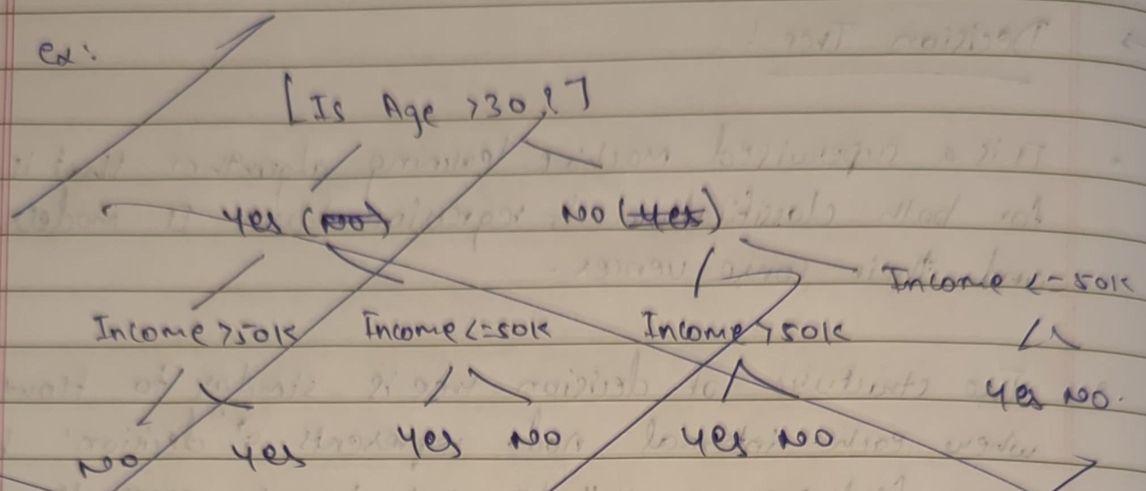
- The structure of decision tree is similar to flowchart. where each internal node represents a 'decision' based on a feature or attribute, each branch represents the outcome of that decision & each leaf node represents a class label or a continous value.

- **Terminologies:**

- Root Node: Represents entire dataset.
- Decision nodes: Represents decision based on a feature's value. (e.g. Age <=30)
- leaf nodes: final nodes that provide the outcome.
- Branches: The paths connecting nodes, representing the decision outcomes at each step.
- features: Attributes used to make decisions at each node.
- splitting criteria: - Information Gain } for classification
  Gini Impurity
  mean squared error (MSE) } for regression.
- Pruning: Process of removing branches to prevent overfitting
- Depth: no. of levels in the tree.

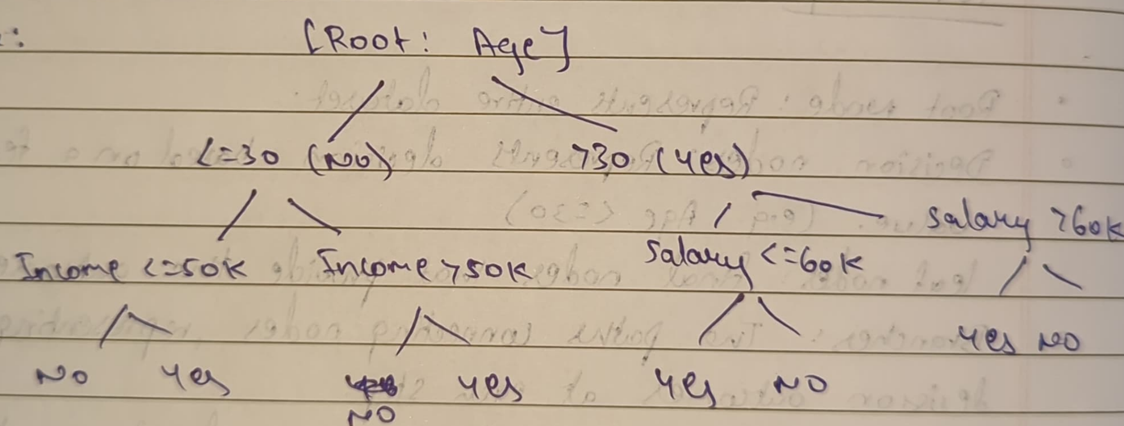
- Diagram of Decision Tree:

ex:



~~This tree predicts whether someone is likely to buy a product based on their age & income.~~

ex:



- Root node asks a decision tree based on Age, & based on whether the condition holds (Age <=30 or >30) the data is further split into diff. features like income & salary, eventually leading to a classification (Yes/No) is made.

- <u>Classification tree:</u>

- The goal is to predict a categorical outcome like 'Yes or No', 'spam' or 'not spam'
- ex: Given a dataset of emails, predict whether an

email is spam or not spam based on features like the sender, subject & content

- splitting criteria → uses Gini impurity & Information Gain to select best features for splitting.

- **Regression tree:**

- Goal is to predict a continous numerical outcome (value) like ~~no~~ house price, temp., etc.
- ex: Given a dataset of house prices, predict the price of a house based on features like square footage, no. of ~~be~~ bedrooms, location, etc
- splitting criteria → uses Mean squared error (MSE)

↳ **Entropy:**

- It is the measure of unpredictability or impurity in a dataset. It is used to quantify the uncertainty in the target variable.
- Entropy helps ~~guide~~ ~~the~~ quantifying the level of uncertainty in the data.

formula :- $H(S) = - \sum_{i=1}^{k} p(C_i) \log_2 p(C_i)$

- **Entropy of a Partition:**

when building a Decision Tree, you split a dataset into subsets based on a feature. The entropy of a Partition refers to the entropy calculated for each subset resulting from the split

↳ Creating a Decision Tree:

i) Collect & Prepare Data
- gather dataset that contains both input features & target variables.

ii) ~~Goh~~ Choose a splitting criterion:
It helps determine which feature to use for the split.
a) Info. gain (IG): It measures how well a feature separates the data based on entropy.
b) Gini Index: It measures the impurity of the dataset, with 0 being pure.

iii) calculate the best split:
- Info. gain (IG): The feature with highest information gain is chosen for the split.
- Gini Index: The feature with lowest Gini index is selected for the split.

iv) split the data into subsets.
v) Stopping criteria:
The recursion stops here.

↳ Random forests:

- It is an ensemble learning technique that combines multiple decision trees to improve the performance of the model.

- Commonly used for both classification & regression tasks.

- steps - Training data
  - Bootstrapping (create several subsets)
  - Building trees
  - voting
  - Majority vote.

- Adv: - Reduces overfitting
  - Handles missing data
  - Robust to noise
  - versatility.

- Disadv: - ~~expensive~~ computationally expensive
  - Slower predictions.

↳ Neural Networks:

- Neural networks are class of machine learning models inspired by the way the human brain processes information.
- Neural networks are used for a variety of tasks such as classification, regression & even deep learning for more complex applications.

i) Perceptrons:
- It forms the foundation for more complex neural networks
- single layer neural network designed to classify data into two classes.

- Components?
a) Input layer: consists of input features
b) weights: each input is associated with a weight that signifies its importance.

c) Bias: A bias term is added to the weighted sum to allow the model to make predictions even when all inputs are zero.
d) Activation function: Determines the output (0 or 1)

- Algorithm - initialize weights
  - compute output
  - update weights
  - Repeat.

ii) <u>Feed forward Neural Networks!</u>

- more complex neural networks that consists of multiple layers of neurons (nodes). unlike perceptron, FNN can learn non-linear relationships in data.
- <u>components:</u>

a) Input layer: This layer recieves the input features for the model.
b) Hidden layers: These layers perform transformations of the inputs, enabling the network to learn complex patterns.
c) output layer: Produces final output of the model, which can be used for classification & regression
d) weights & biases: each connection betn neurons has an associated weight & each neuron has a bias term

- forward Propagation: Data flows in one direction (from input layer to output layer)

steps:- Input
- weighted sum

- Activation
- Repeat
- output.

- Activation function:

• This functn introduces non linearity, allowing neural networks to model more complex patterns.
• commonly used activation functions - sigmoid
  - Tanh
  - ReLU

- FNN Adv:
• can handle both classification & regression problems.
• can model ~~non linear~~ complex relationships betn inputs & outputs.
• layered architecture.
• customizable.

- Disadv:
• overfitting (especially with small datasets)
• computationally expensive
• Require more data

iii) Backpropagation:

• Algorithm used to train neural networks by updating the weights & Biases based on the error betn predicted & actual output.

- steps:
i) Forward propagation: calculate the output using

current weights & biases

ii) calculate error: compute the error betn predicted output & actual output

for classification → cross entropy loss

for regression → MSE

iii) Backward pass (compute gradients): use the chain rule of calculus to calculate the gradients of the error with respect to each weight in the network.

iv) update weights & Biases: update the weights & Biases using algorithm like Gradient descent.

v) Repeat: Repeat the process for multiple epochs until the error converges to an acceptable level.

- Adv:
  - Efficient training
  - Adaptability (works with a wide range of tasks)
  - scalable.

- Disadv:
  - vanishing gradients (gradients can become too small or too large, making training unstable)
  - slow convergence: Takes a lot of time for training deep networks.
  - overfitting risks.

↳ why mapreduce:

- The core idea is to use a divide & conquer strategy to achieve parallelization & distribution of work across a cluster of machines making it highly scalable & ~~tolerant~~ fault tolerant
- Reasons - scalability

- parallel processing
- fault Tolerance
- simplicity
- High throughput

↳ **word count example:**

Given a large ~~count~~ collection of documents, the goal is to count the frequency of each word in the entire dataset.

- Steps:

i) Input: collection of documents.

ii) Map phase: - each mapper processes a portion of the input data.
- for each word in the document, the mapper emits a key value pair.

iii) Shuffle & sort: The system sorts the key value pairs by key & groups them together.

iv) Reduce phase: Reducer aggregates the values for each key.

v) output: The final output is the count of each word in the entire collection of documents.

↳ **Matrix Multiplication example:**

Here, map reduce ~~is~~ can be very useful, especially for large matrices that cant fit into the memory of a single machine.

- Steps:

i) Input: Two matrices A & B
A is divided into rows & B into columns.

ii) Map phase: -Each mapper processes a row from matrix A & a column from matrix B.
-for each pair of row from matrix A & column from B, the mapper computes the dot product for one element of the resulting matrix C.

iii) shuffle & sort: The system groups the key value pair by the position in the result matrix.

iv) Reduce: reducer sums the values for each key to compute final value of each element in C.

v) output: The final result is the matrix C.