1.) Explain the following with examples: a. Decision Tree, b. Decision Tree Learning c. Decision Tree Representation.

a. Decision Tree

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks.

They are very powerful algorithms, capable of fitting complex datasets.

A Decision Tree is a flowchart-like structure in machine learning that represents a series of decisions to arrive at a specific outcome.

It works by recursively partitioning the data into subsets based on the feature that provides the best split. This process continues until a stopping criterion is met, such as reaching a maximum depth or no further improvement in purity.

It's similar to how humans make decisions by considering various factors and choosing the path that aligns best with their goal.

b. decision Tree Learning

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.

Decision tree learning is the process of constructing a decision tree from training data.

It involves selecting the best features to split on at each node and determining the optimal partitioning criteria.

Example: Suppose you have a dataset of emails labeled as spam or not spam. Decision tree learning would involve selecting features like the presence of certain keywords, the length of the email, or the frequency of punctuation marks to split the data and build a tree that can accurately classify emails as spam or not spam based on those features.

c. Decision Tree Representation

•Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

• Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

• An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node.

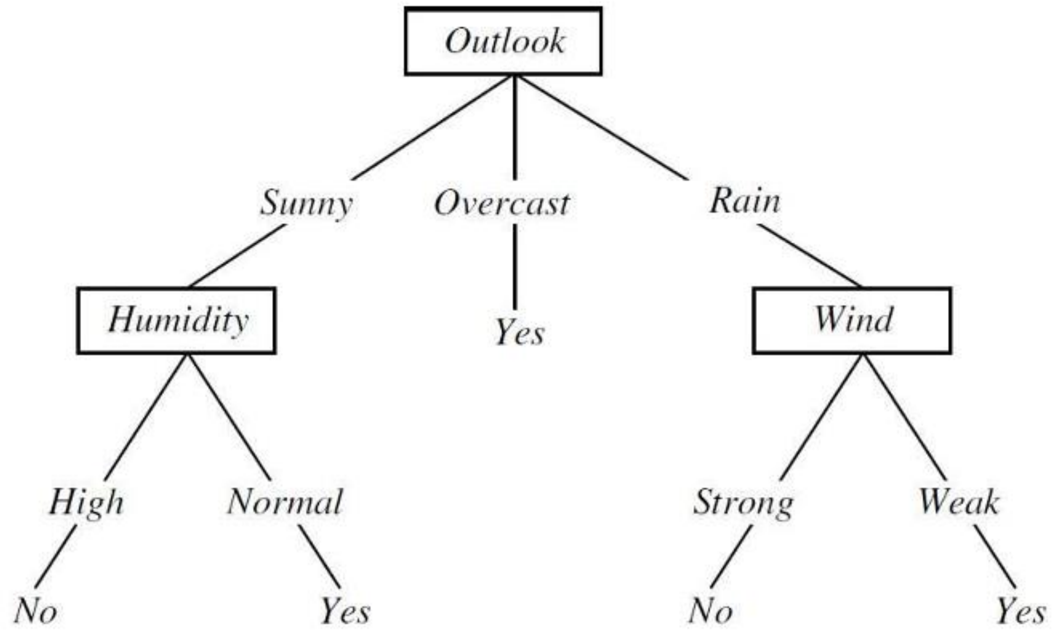


FIGURE: A decision tree for the concept Play Tennis. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions

For example, the decision tree shown in above figure corresponds to the expression

(Outlook = Sunny A Humidity = Normal)
V (Outlook = Overcast)
V (Outlook = Rain A Wind = Weak)

2.) Describe the ID3 algorithm for decision tree learning with example.

• The ID3 (Iterative Dichotomiser 3) algorithm is a popular decision tree learning algorithm used to construct decision trees from a dataset.

• It was developed by Ross Quinlan in 1986.

• It is used to generate a decision tree from a given data set by employing top-down greedy search to test each attribute at every node of the tree.

• The resulting tree is used to classify future examples.

•The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree.

•ID3 uses information gain measure to select among the candidate attributes at each step while growing the tree.

Steps of the ID3 Algorithm

1. Calculate the entropy and information gain of every attribute in the dataset.

$$Entropy\ (S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Where, p+ is the proportion of positive examples in S

p– is the proportion of negative examples in S.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

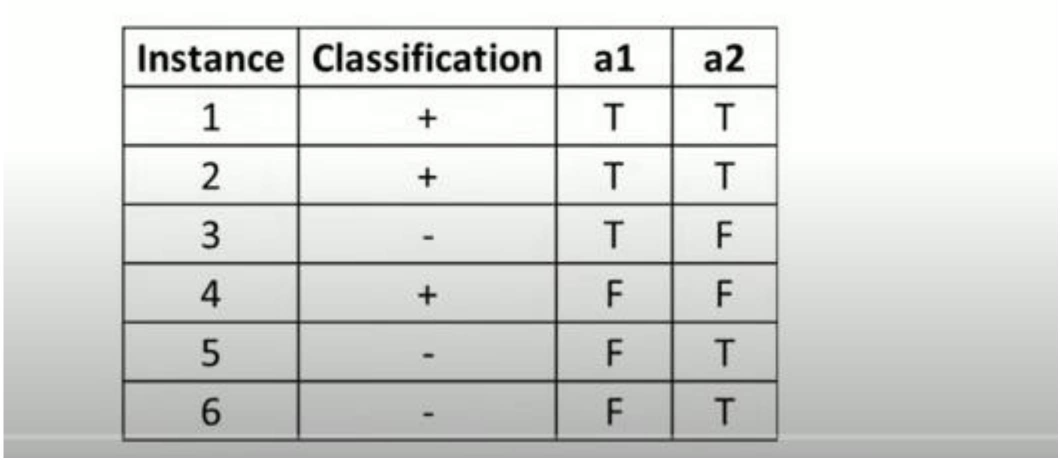
where Values(A) is the set of all possible values for attribute A, and S, is the subset of S for which attribute A has value v (i.e., S_v= {s ∈ S|A(s) = v})

2. split the dataset into subsets using the attribute for which entropy is minimum i.e., Gain is maximum.

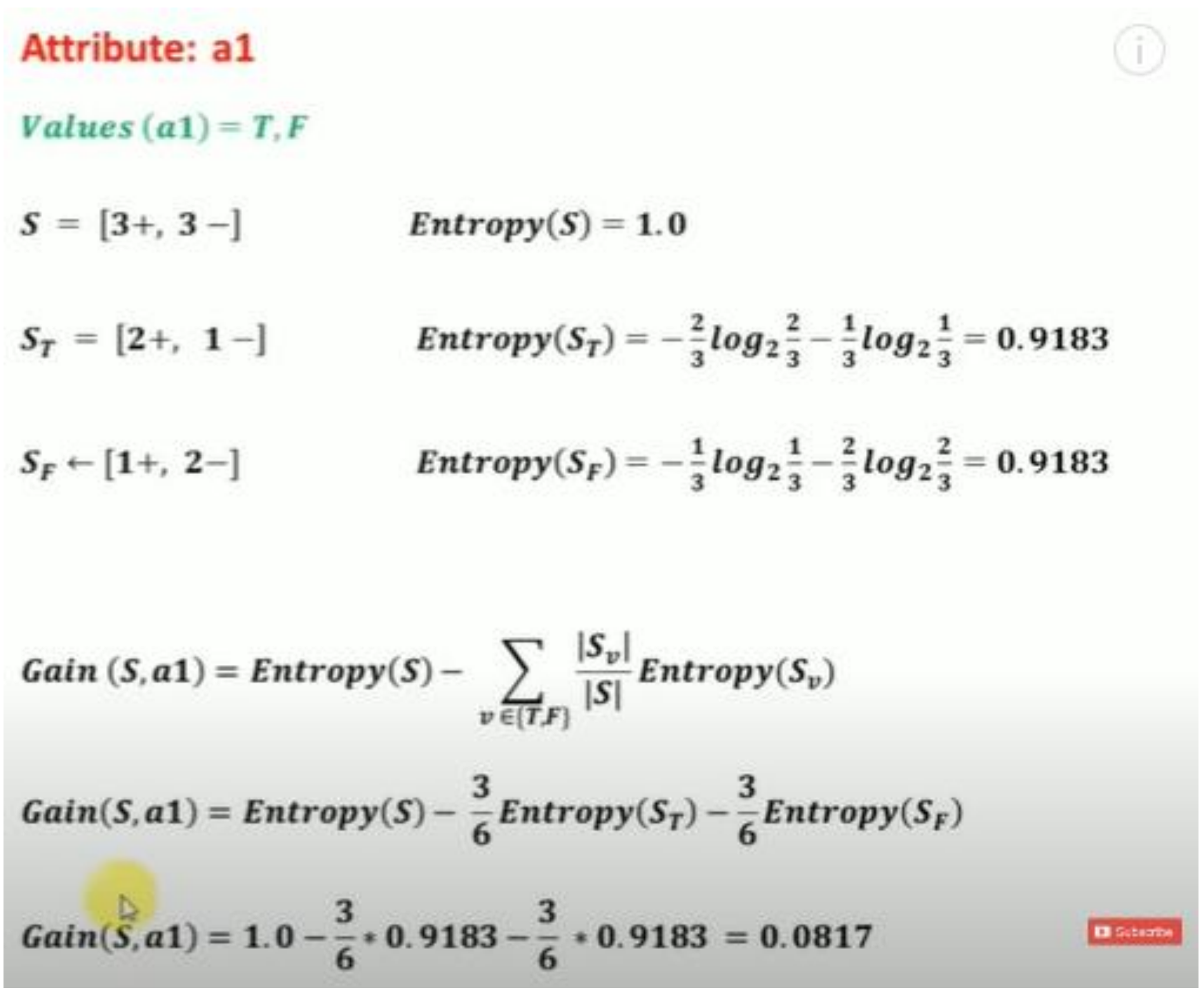
3. Make a decision tree root node with that attribute.

4. Recurse on subsets using remaining attributes.

EXAMPLE:



Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T



Attribute: a1

$Values\ (a1) = T, F$

$S = [3+, 3-]$ $\qquad$ $Entropy(S) = 1.0$

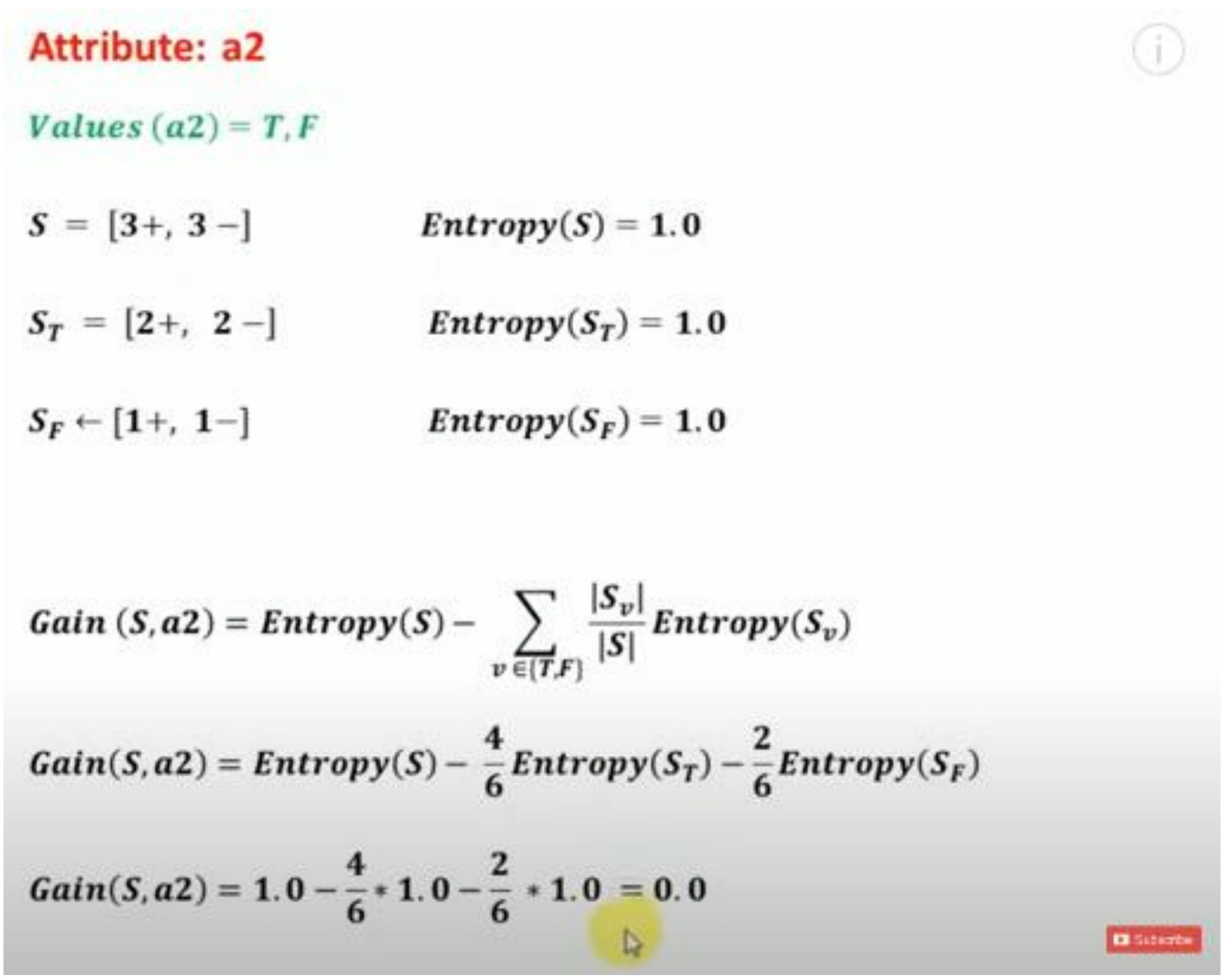
$S_T = [2+, 1-]$ $\qquad$ $Entropy(S_T) = -\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3} = 0.9183$

$S_F \leftarrow [1+, 2-]$ $\qquad$ $Entropy(S_F) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.9183$

$$Gain\ (S, a1) = Entropy(S) - \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, a1) = Entropy(S) - \frac{3}{6}Entropy(S_T) - \frac{3}{6}Entropy(S_F)$$

$$Gain(S, a1) = 1.0 - \frac{3}{6} * 0.9183 - \frac{3}{6} * 0.9183 = 0.0817$$



Attribute: a2

$Values\ (a2) = T, F$

$S = [3+, 3-]$ $\qquad$ $Entropy(S) = 1.0$

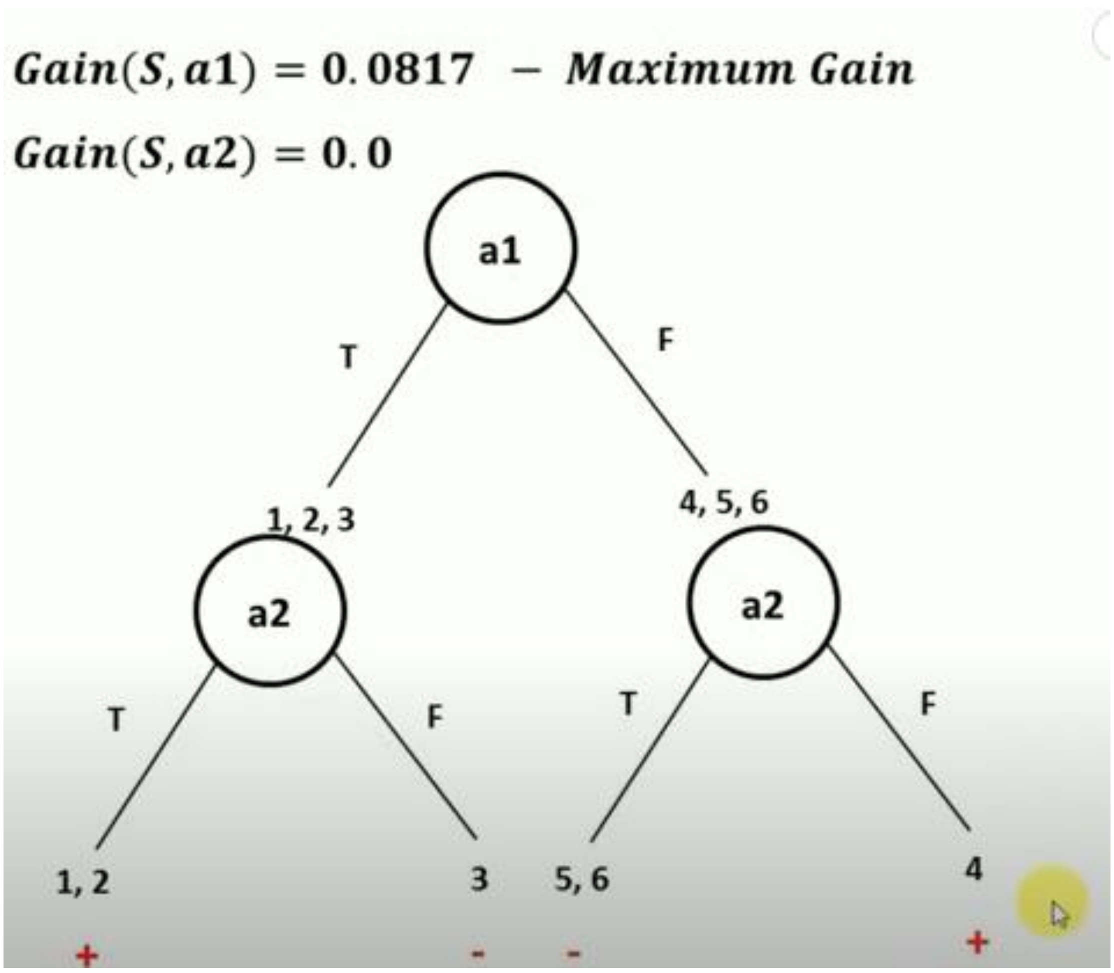
$S_T = [2+, 2-]$ $\qquad$ $Entropy(S_T) = 1.0$

$S_F \leftarrow [1+, 1-]$ $\qquad$ $Entropy(S_F) = 1.0$

$$Gain\ (S, a2) = Entropy(S) - \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, a2) = Entropy(S) - \frac{4}{6}Entropy(S_T) - \frac{2}{6}Entropy(S_F)$$

$$Gain(S, a2) = 1.0 - \frac{4}{6} * 1.0 - \frac{2}{6} * 1.0 = 0.0$$



3.) What do you mean by Gain and Entropy? How is it used to build the Decision tree In algorithm? Illustrate using an example.

ENTROPY:

• Entropy measures the impurity of a collection of examples.

• Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

$$Entropy\ (S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Where, p+ is the proportion of positive examples in S

p– is the proportion of negative examples in S.

• The entropy is 0 if all members of S belong to the same class .

• The entropy is 1 when the collection contains an equal number of positive and negative examples .

• If the collection contains unequal numbers of positive and negative examples, the entropy is between 0 and 1.

Example:

Suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples. Then the entropy of S relative to this boolean classification is

$$\begin{aligned} Entropy([9+,5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

GAIN:

Information gain measures how well a given attribute separates the training examples according to their target classification.

The information gain is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

The information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as,

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where Values(A) is the set of all possible values for attribute A, and S, is the subset of S for which attribute A has value v (i.e., S_v= {s ∈ S|A(s) = v})

The attribute with the highest information gain is chosen as the splitting attribute at each step of the algorithm**.**

For example, suppose S is a collection of training-example days described by attributes including *Wind*, which can have the values *Weak* or *Strong*. As before, assume S is a collection containing 14 examples, [9+, 5−]. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have *Wind* = *Weak*, and the remainder have *Wind* = *Strong*. The information gain due to sorting the original 14 examples by the attribute *Wind* may then be calculated as

$$\begin{aligned} Values(Wind) &= Weak, Strong \\ S &= [9+, 5-] \\ S_{Weak} &\leftarrow [6+, 2-] \\ S_{Strong} &\leftarrow [3+, 3-] \\ Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\ &= Entropy(S) - (8/14) Entropy(S_{Weak}) \\ &\quad - (6/14) Entropy(S_{Strong}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

• ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

4.) What are issues in decision tree learning? Explain briefly How are they overcome?

Issues in Decision Tree Learning

1. Overfitting: Decision trees can become overly complex and fit the training data too closely, leading to poor performance on unseen data. This is especially true for noisy datasets or when the tree grows very deep.

2. Sensitivity to irrelevant features: The algorithm might choose irrelevant features for splitting if they happen to correlate with the target variable by chance in the training data. This can reduce the interpretability and generalizability of the tree.

3. Bias: Decision trees can inherit biases from the training data, leading to discriminatory or unfair outcomes. For example, if the training data contains more samples from one class, the tree might favor that class in its predictions.

4. Missing values: Decision trees typically handle missing values by imputing them with mean/mode or creating separate branches, which can introduce noise and affect tree structure.

5. Computational complexity: Building large decision trees can be computationally expensive, especially for massive datasets.

Overcoming these issues:

1. Pruning: Techniques like pre-pruning (stopping tree growth early) or post-pruning (removing unnecessary branches) can help prevent overfitting.

2. Regularization: Methods like setting a maximum depth or using penalties for complex trees can discourage overfitting and encourage simpler, more generalizable models.

3. Feature selection: Carefully selecting relevant features before building the tree can reduce the impact of irrelevant features and improve interpretability.

4. Data balancing: Techniques like oversampling or undersampling can address class imbalance and reduce bias.

5. Handling missing values: More sophisticated methods like imputation by k-nearest neighbors or decision trees specifically designed for missing values can improve handling of missing data.

6. Ensemble methods: Combining multiple decision trees into an ensemble (e.g., random forest) can improve accuracy and robustness by averaging predictions and reducing overfitting.

5,) Define the Bayesian theorem? What is the relevance and features of the Bayesian theorem? Explain the practical difficulties of the Bayesian theorem.

- Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

- It is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- Bayes theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability $P(h|D)$, from the prior probability $P(h)$, together with $P(D)$ and $P(D|h)$.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Where, P(h|D) is the probability of hypothesis h given the data D. This is called the posterior probability.

P(D|h) is the probability of data d given that the hypothesis h was true.

P(h) is the probability of hypothesis h being true. This is called the prior probability of h.

P(D) is the probability of the data. This is called the prior probability of D.

- P(h|D) increases with P(h) and with P(D|h) according to Bayes theorem.
- P(h|D) decreases as P(D) increases, because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

<u>Features of Bayesian Learning Methods</u>

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting

  (1) a prior probability for each candidate hypothesis, and

  (2) a probability distribution over observed data for each possible hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

<u>Practical difficulty in applying Bayesian methods:</u>

1. One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.

2. A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case. In certain specialized situations, this computational cost can be significantly reduced.

6.) Explain Naïve Bayes Classifier with a simple Example.

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?
The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem

Example:

Here's how the Naïve Bayes Classifier works with a simple example of classifying fruits as either "Apple" or "Orange" based on two features: colour and shape.

Suppose we have the following training dataset:

Fruit	**Colour**	**Shape**
Apple	Red	Round
Apple	Red	Round
Orange	Orange	Round
Orange	Orange	Round
Orange	Orange	Oval

Now, let's say we have a new fruit with the following features: Color = Red and Shape = Round, and we want to classify it using the Naïve Bayes Classifier.

Step 1: Calculate Prior Probabilities:

- $P(\text{Apple}) = \frac{\text{Number of Apples}}{\text{Total Number of Fruits}} = \frac{2}{5} = 0.4$
- $P(\text{Orange}) = \frac{\text{Number of Oranges}}{\text{Total Number of Fruits}} = \frac{3}{5} = 0.6$

Step 2: Calculate Likelihood Probabilities:

- For each feature (Colour and Shape), calculate the conditional probabilities given each class (Apple or Orange).
- Since we assume feature independence, we calculate these probabilities separately for each feature.
- For Colour = Red:

  - $P(\text{Red}|\text{Apple}) = \frac{\text{Number of Red Apples}}{\text{Total Number of Apples}} = \frac{2}{2} = 1.0$
  - $P(\text{Red}|\text{Orange}) = \frac{\text{Number of Red Oranges}}{\text{Total Number of Oranges}} = \frac{0}{3} = 0.0$

- For Shape = Round:

  - $P(\text{Round}|\text{Apple}) = \frac{\text{Number of Round Apples}}{\text{Total Number of Apples}} = \frac{2}{2} = 1.0$
  - $P(\text{Round}|\text{Orange}) = \frac{\text{Number of Round Oranges}}{\text{Total Number of Oranges}} = \frac{2}{3} \approx 0.667$

Step 3: Calculate Posterior Probabilities:

- Using Bayes' theorem, calculate the posterior probabilities for each class.

- For Apple:
  $P(\text{Apple}|\text{Red}, \text{Round}) = P(\text{Red}|\text{Apple}) \times P(\text{Round}|\text{Apple}) \times P(\text{Apple}) = 1.0 \times 1.0 \times 0.4 = 0.4$
- For Orange:
  $P(\text{Orange}|\text{Red}, \text{Round}) = P(\text{Red}|\text{Orange}) \times P(\text{Round}|\text{Orange}) \times P(\text{Orange}) = 0.0 \times 0.667 \times 0.6 = 0$

Step 4: Make Prediction:

- Since P(Apple|Red, Round) > P(Orange|Red, Round)P(Apple|Red, Round)>P(Orange|Red, Round), we classify the fruit as an Apple.

In this example, the Naïve Bayes Classifier predicts that the fruit is an Apple based on its color and shape features. Despite its simplicity and the assumption of feature independence, Naïve Bayes can often perform surprisingly well in practice, especially for text classification tasks.

7.) Consider a medical diagnosis problem in which there are two alternative hypotheses: 1. that the patient has a particular form of cancer (+) and 2. That the patient does not (-). A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer. Determine whether the patient has Cancer or not using the MAP hypothesis.

Given,

$P(\text{cancer}) = 0.008$

$P(\neg\text{cancer}) = 0.992$

$P(+ \mid \text{cancer}) = 0.98$

$P(- \mid \text{cancer}) = 0.02$

$P(+ \mid \neg\text{cancer}) = 0.03$

$P(- \mid \neg\text{cancer}) = 0.97$

Bayes theorem

posterior Probability,

$$P(h \mid D) = \frac{P(D \mid h) \cdot P(h)}{P(D)}$$

Case-I

$$P(\text{cancer} \mid +) = \frac{P(+ \mid \text{cancer}) \cdot P(\text{cancer})}{P(+)}$$

$$= (0.98)(0.08) \qquad (\because \text{ignore } P(+))$$

$$= 0.0078$$

$$P(\neg\text{cancer} \mid +) = \frac{P(+ \mid \neg\text{cancer}) \cdot P(\neg\text{cancer})}{P(+)}$$

$$= (0.03)(0.992)$$

$$= 0.0298$$

Case-II

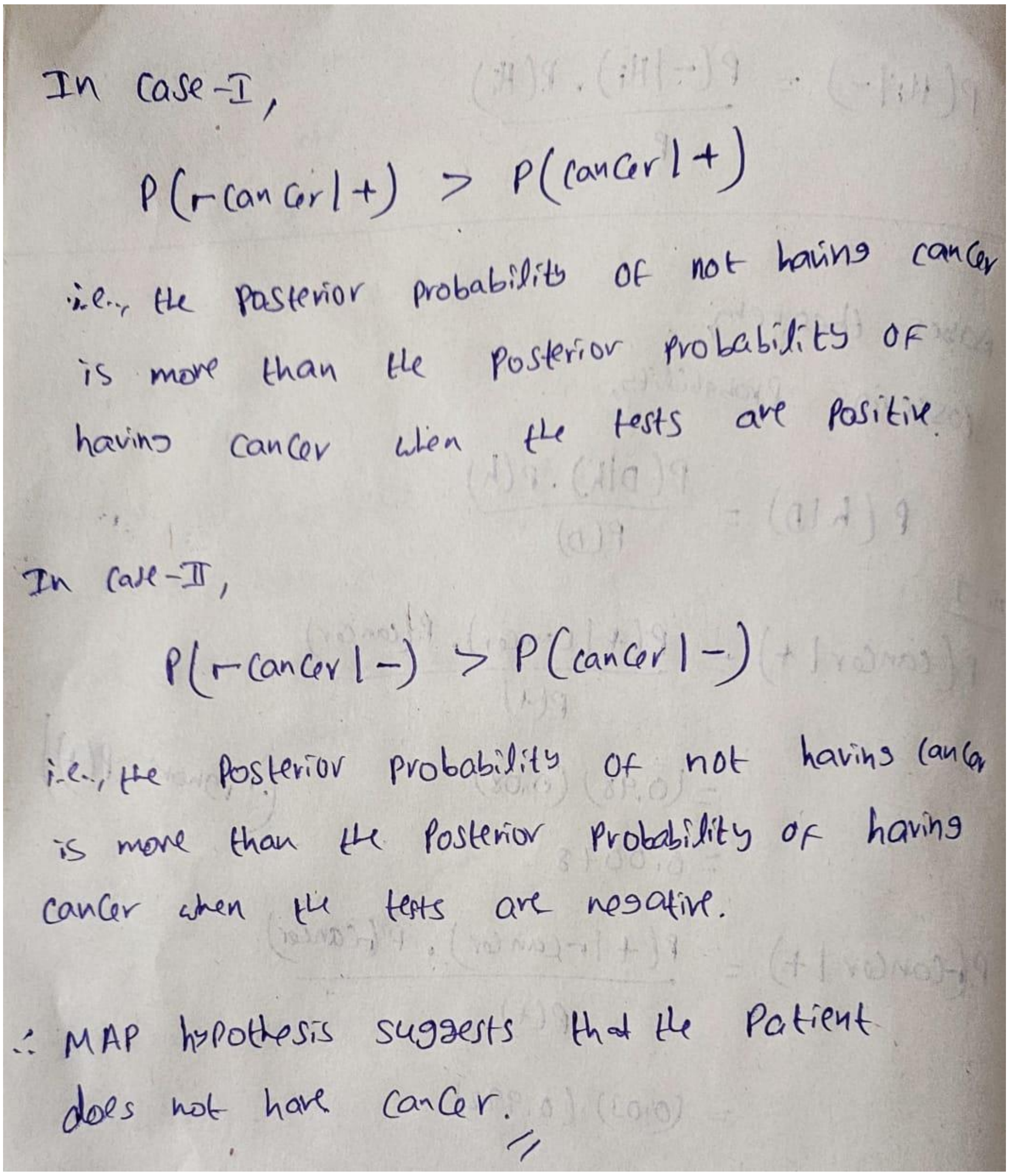
$$P(\text{cancer} \mid -) = \frac{P(- \mid \text{cancer}) \cdot P(\text{cancer})}{P(-)}$$

$$= (0.02)(0.08) \qquad (\because \text{ignore } P(-))$$

$$= 0.00016$$

$$P(\neg\text{cancer} \mid -) = \frac{P(- \mid \neg\text{cancer}) \cdot P(\neg\text{cancer})}{P(-)}$$

$$= (0.97)(0.992) = 0.96224$$



In case-I,

$$P(\neg cancer|+) > P(cancer|+)$$

i.e., the posterior probability of not having cancer is more than the posterior probability of having cancer when the tests are positive.

In case-II,

$$P(\neg cancer|-) > P(cancer|-)$$

i.e., the posterior probability of not having cancer is more than the posterior probability of having cancer when the tests are negative.

∴ MAP hypothesis suggests that the patient does not have cancer. //

8.) Describe K-nearest Neighbour learning Algorithm for continues (real) valued target function.

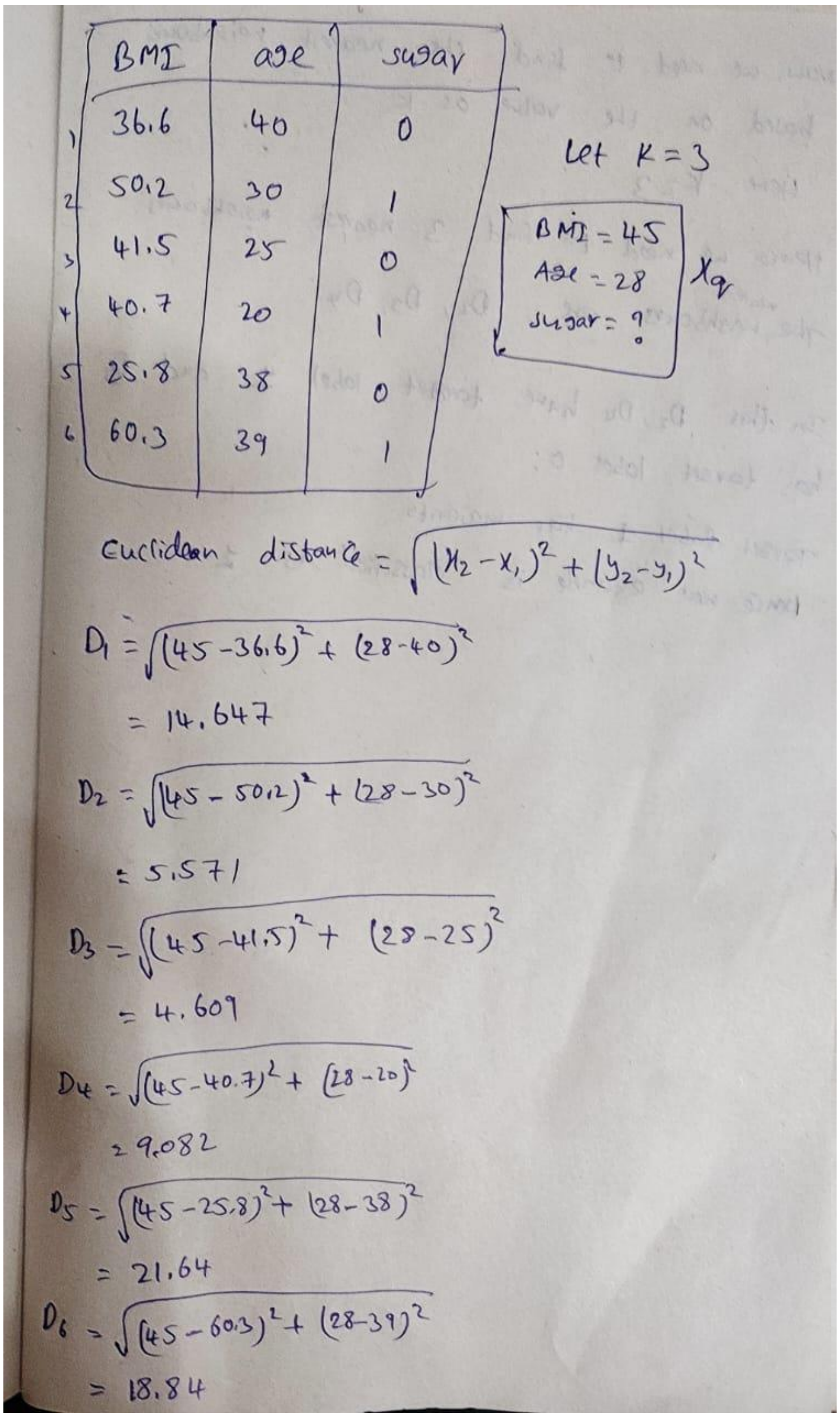
- It is a supervised machine learning algorithm.
- The algorithm can be used to solve both classification and regression problem statements.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.
- KNN calculates the distance from all points in the proximity of the unknown data and filters out the ones with the shortest distances to it. As a result, it's often referred to as a distance-based algorithm.

• In order to correctly classify the results, we must first determine the value of K (Number of Nearest Neighbours).

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbours
- **Step-2:** Calculate the Euclidean distance of **K number of neighbours**
- **Step-3:** Take the K nearest neighbours as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbours, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbour is maximum.
- **Step-6:** Our model is ready.



	BMI	age	sugar
1	36.6	40	0
2	50.2	30	1
3	41.5	25	0
4	40.7	20	1
5	25.8	38	0
6	60.3	39	1

let $K = 3$

BMI = 45
Age = 28
sugar = ?
X_q

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

$D_1 = \sqrt{(45 - 36.6)^2 + (28 - 40)^2}$
$= 14.647$

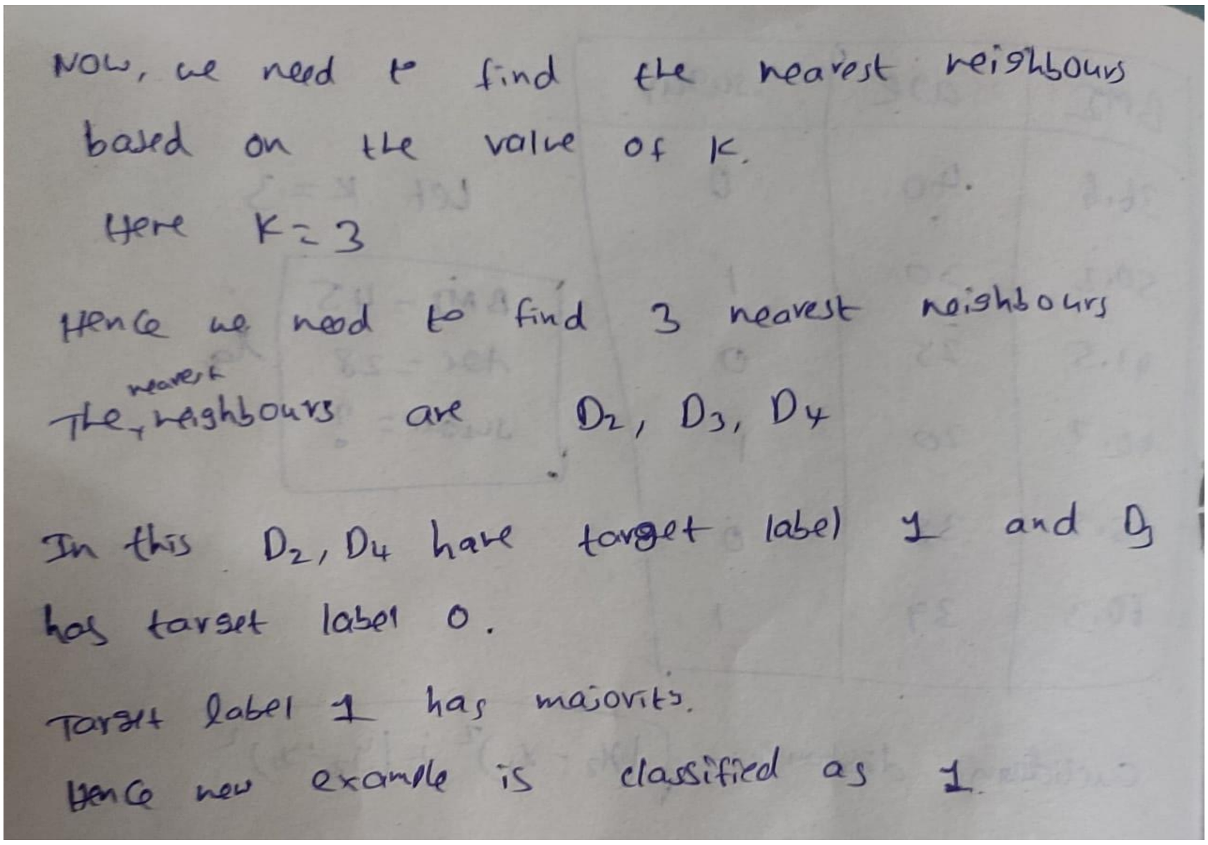
$D_2 = \sqrt{(45 - 50.2)^2 + (28 - 30)^2}$
$= 5.571$

$D_3 = \sqrt{(45 - 41.5)^2 + (28 - 25)^2}$
$= 4.609$

$D_4 = \sqrt{(45 - 40.7)^2 + (28 - 20)^2}$
$= 9.082$

$D_5 = \sqrt{(45 - 25.8)^2 + (28 - 38)^2}$
$= 21.64$

$D_6 = \sqrt{(45 - 60.3)^2 + (28 - 39)^2}$
$= 18.84$



Now, we need to find the nearest neighbours based on the value of K.

Here K = 3

Hence we need to find 3 nearest neighbours

The nearest neighbours are D_2, D_3, D_4

In this D_2, D_4 have target label 1 and D_3 has target label 0.

Target label 1 has majority.

Hence new example is classified as 1.

9.) Discuss the major drawbacks of K-nearest Neighbour learning Algorithm and how it can be corrected

1. **Computational Complexity:**
   - As the dataset grows, the computational complexity of KNN increases significantly because it requires calculating distances between the query point and all training data points.
   - **Correction:** Several methods can help mitigate computational complexity, such as:
     - Using approximate nearest neighbor algorithms like KD-trees or Ball trees for efficient nearest neighbor search.
     - Reducing the dimensionality of the feature space through techniques like Principal Component Analysis (PCA) or feature selection to speed up computation.
2. **Storage Requirements:**
   - KNN requires storing the entire training dataset, which can be memory-intensive for large datasets, especially when dealing with high-dimensional data.
   - **Correction:** Techniques to reduce storage requirements include:
     - Using data structures like KD-trees or Ball trees for efficient storage and retrieval of nearest neighbors.
     - Applying dimensionality reduction techniques like PCA to reduce the number of features and hence the storage requirements.
3. **Slow Prediction Phase:**

- During the prediction phase, KNN needs to calculate distances between the query point and all training data points, resulting in slow prediction times, especially for large datasets.
- **Correction:** Apart from approximate nearest neighbor algorithms and dimensionality reduction techniques mentioned earlier, other strategies include:
    - Implementing parallelization to distribute the workload across multiple processors or machines.
    - Using specialized hardware such as GPUs (Graphics Processing Units) for faster distance computations.

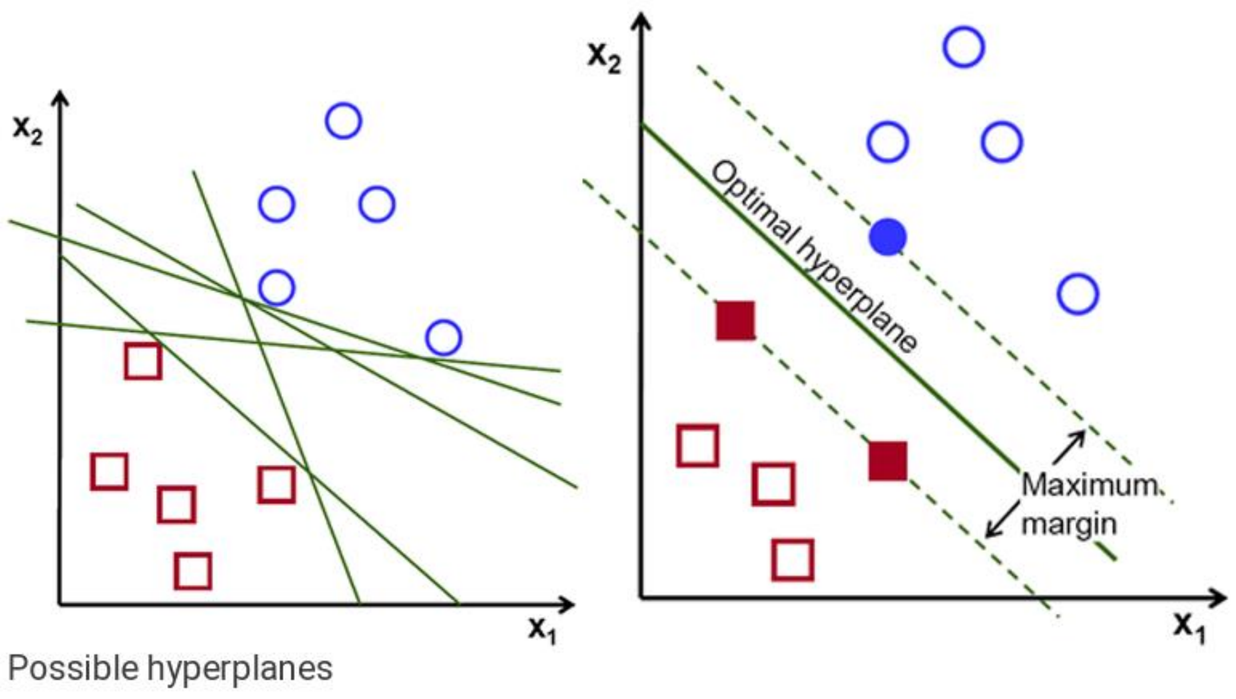
4. **Sensitivity to Irrelevant Features:**
    - KNN treats all features equally, which means it can be sensitive to irrelevant or noisy features in the dataset, leading to poor performance.
    - **Correction:** To address this issue:
        - Feature selection techniques can be applied to remove irrelevant features and reduce the dimensionality of the feature space.
        - Feature engineering methods can be used to create new features that are more informative for the task at hand.
        - Regularization techniques can be incorporated to penalize complex models and reduce the influence of noisy features.
5. **Optimal Selection of K:**
    - The choice of K significantly impacts the performance of the KNN algorithm, and selecting an optimal K value is often challenging.
    - **Correction:** Strategies to address this issue include:
        - Using cross-validation techniques to evaluate the performance of different K values and select the one that yields the best results on validation data.
        - Employing grid search or randomized search algorithms to systematically explore the space of possible K values and find the optimal one.

10.) Explain Linear SVM Classifier with a simple Example.

• A Linear Support Vector Machine (SVM) is a machine learning algorithm used for classification tasks.

• The objective of the support vector machine algorithm is to find a hyperplane(a straight line in 2D or a flat plane in higher dimensions) in an N-dimensional space(N — the number of features) that distinctly classifies the data points.
• To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e., the maximum distance between data points of both classes.

• Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.



Support vectors are data points that are closer to the hyperplane and influence the position and 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

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier

Hyper plane is represented by the equation

$$y = w^* X + b$$

Where, w is the weight vector,

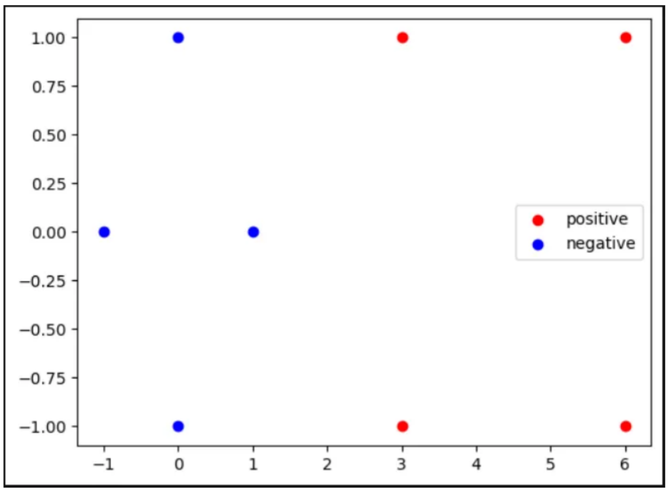
X is the feature vector,

b is the bias term.

Example:

Positively labelled data points (3,1)(3,-1)(6,1)(6,-1) and Negatively labelled data points (1,0)(0,1)(0,-1)(-1,0)

Solution: for all negative labelled output is -1 and for all positive labelled output is 1.



graph for above table

Now adding 1 to all points

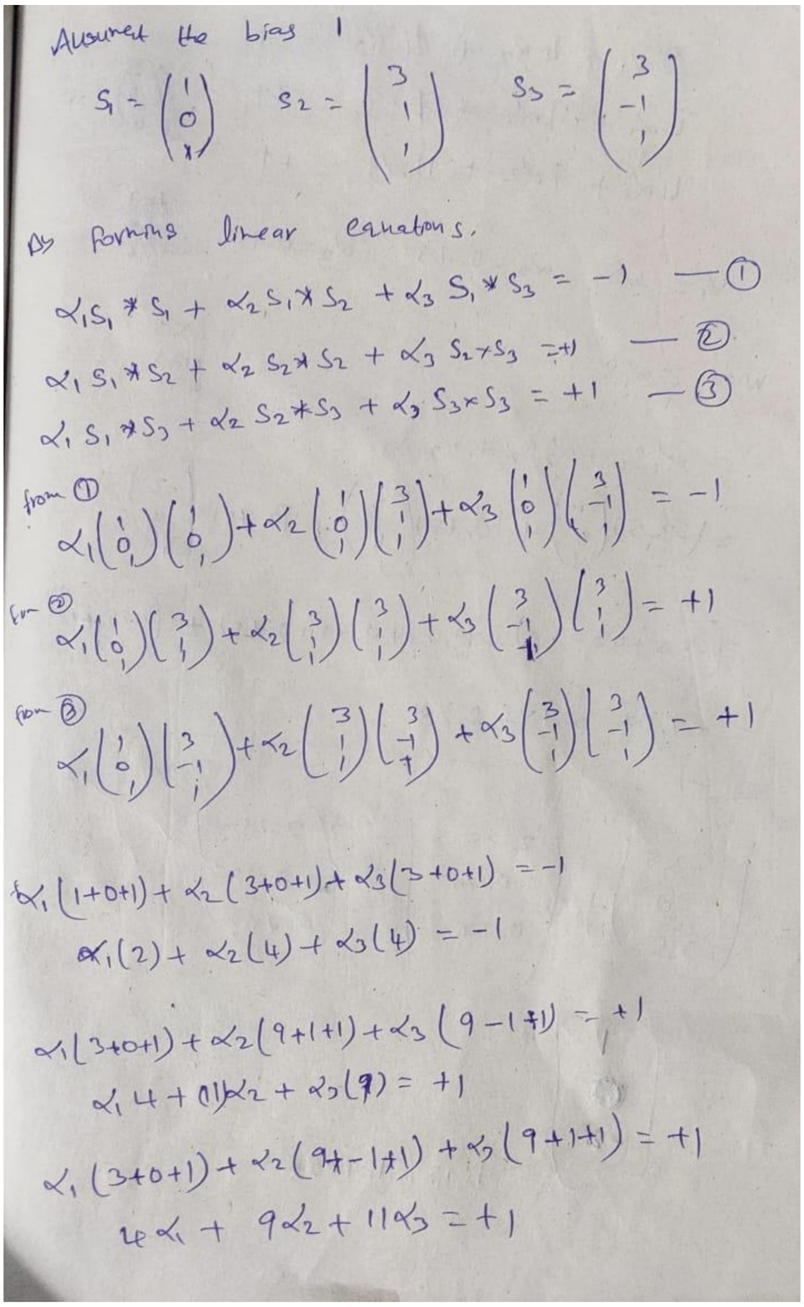
$s_1 = (3,1) \Rightarrow s_1` = (3,1,1)$| $s_2 = (3,-1) \Rightarrow s_2` = (3,1,-1)$

$s_3 = (6,1) \Rightarrow s_3` = (6,1,1)$ |$s_4 = (6,-1) \Rightarrow s_4` = (6,1,-1)$

$s_5 = (1,0) \Rightarrow s_5` = (1,0,1)$ |$s_6 = (0,1) \Rightarrow s_6` = (0,1,1)$

$s_7 = (0,-1) \Rightarrow s_7 = (0,-1,1)$ |$s_8 = (-1,0) \Rightarrow s_8` = (-1,0,1)$

from the graph we can see there is one negative point $s_1 = (1,0,1)$ and two positive points $s_2 = (3,1,1)$ & $s_3 = (3,-1,1)$ form support vectors



Assumed the bias 1

$S_1 = \begin{pmatrix}1\\0\\1\end{pmatrix} \quad S_2 = \begin{pmatrix}3\\1\\1\end{pmatrix} \quad S_3 = \begin{pmatrix}3\\-1\\1\end{pmatrix}$

By forming linear equations.

$\alpha_1 S_1 * S_1 + \alpha_2 S_1 * S_2 + \alpha_3 S_1 * S_3 = -1$ — ①

$\alpha_1 S_1 * S_2 + \alpha_2 S_2 * S_2 + \alpha_3 S_2 * S_3 = +1$ — ②

$\alpha_1 S_1 * S_3 + \alpha_2 S_2 * S_3 + \alpha_3 S_3 * S_3 = +1$ — ③

from ①

$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_2\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} = -1$

from ②

$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_2\begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} = +1$

from ③

$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} + \alpha_2\begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} = +1$

$\alpha_1(1+0+1) + \alpha_2(3+0+1) + \alpha_3(3+0+1) = -1$

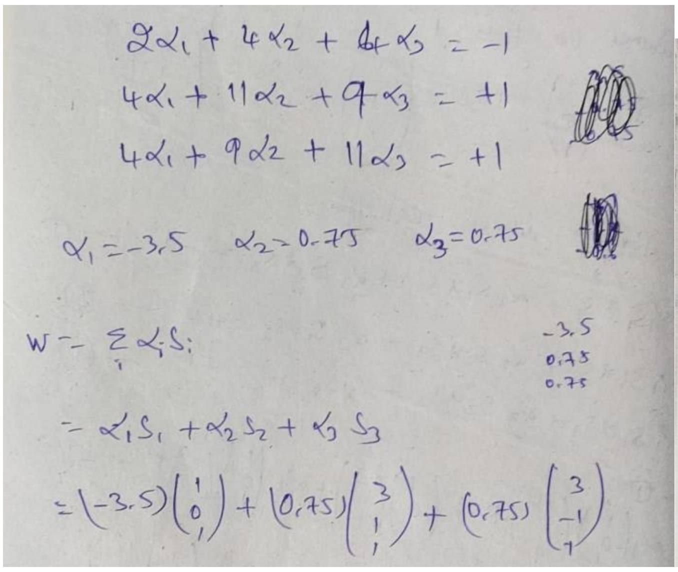
$\alpha_1(2) + \alpha_2(4) + \alpha_3(4) = -1$

$\alpha_1(3+0+1) + \alpha_2(9+1+1) + \alpha_3(9-1+1) = +1$

$\alpha_1 4 + (11)\alpha_2 + \alpha_3(9) = +1$

$\alpha_1(3+0+1) + \alpha_2(9-1+1) + \alpha_3(9+1+1) = +1$

$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = +1$



$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$

$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = +1$

$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = +1$

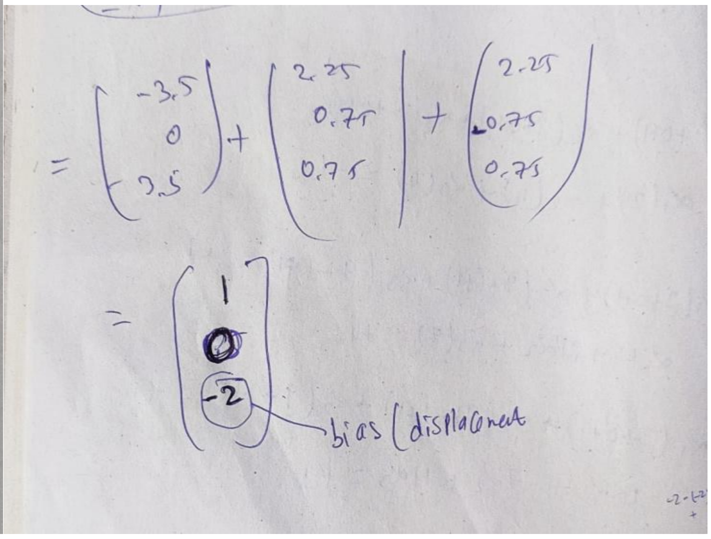
$\alpha_1 = -3.5 \quad \alpha_2 = 0.75 \quad \alpha_3 = 0.75$

-3.5
0.75
0.75

$W = \sum_i \alpha_i S_i$

$= \alpha_1 S_1 + \alpha_2 S_2 + \alpha_3 S_3$

$= (-3.5)\begin{pmatrix}1\\0\\1\end{pmatrix} + (0.75)\begin{pmatrix}3\\1\\1\end{pmatrix} + (0.75)\begin{pmatrix}3\\-1\\1\end{pmatrix}$

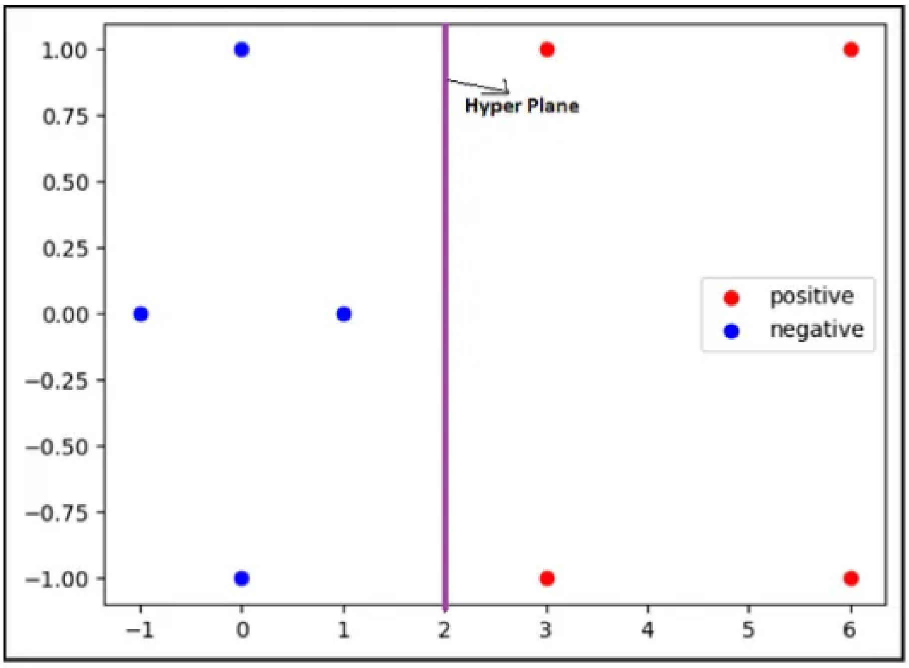


$= \begin{pmatrix}-3.5\\0\\-3.5\end{pmatrix} + \begin{pmatrix}2.25\\0.75\\0.75\end{pmatrix} + \begin{pmatrix}2.25\\-0.75\\0.75\end{pmatrix}$

$= \begin{pmatrix}1\\0\\\boxed{-2}\end{pmatrix}$ bias (displacement y = Wx + b

W = (1,0) and b = 2

so the best fit line or hyper plane is at (0,2) which splits the data points into two classes


Hyper Plane
positive
negative