UNIT III SUPERVISED LEARNING

Linear Regression -Relation between two variables – Steps – Evaluation – Logistic Regression – Decision Tree – Algorithms – Construction – Classification using Decision Tree – Issues – Rule-based Classification – Pruning the Rule Set – Support Vector Machines – Linear SVM – Optimal Hyperplane – Radial Basis Functions – Naïve Bayes Classifier – Bayesian Belief Networks

Linear Regression

Problems with Solutions

Review

If the plot of n pairs of data (x , y) for an experiment appear to indicate a "linear relationship" between y and x, then the method of least squares may be used to write a linear relationship between x and y.
The least squares regression line is the line that minimizes the sum of the squares (d1 + d2 + d3 + d4) of the vertical deviation from each data point to the line (see figure below as an example of 4

points).

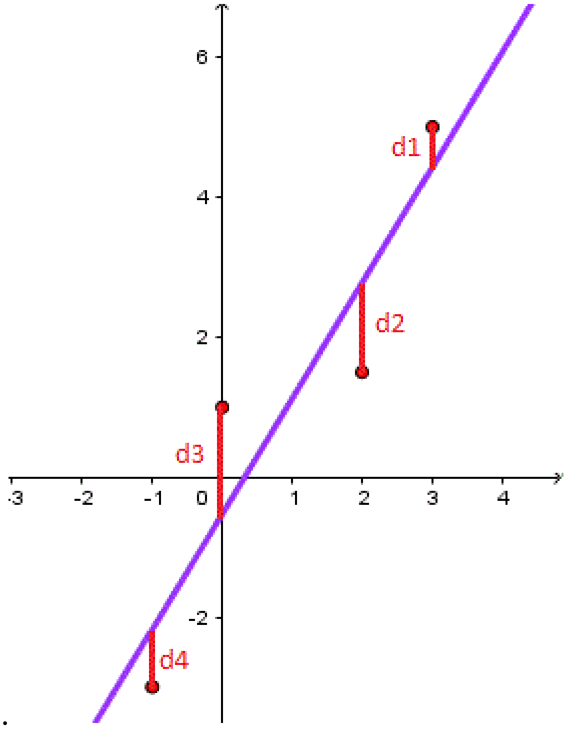


Figure 1. Linear regression where the sum of vertical distances d1 + d2 + d3 + d4 between observed and predicted (line and its equation) values is minimized.The least square regression line for the set of n data points is given by the equation of a line in slope intercept form:

y = a x + b

where a and b are given by

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$b = \frac{1}{n}(\sum_{i=1}^{n} y_i - a\sum_{i=1}^{n} x_i)$$

Figure 2. Formulas for the constants a and b included in the linear regression .

- **Problem 1**

  Consider the following set of points: {(-2 , -1) , (1 , 1) , (3 , 2)}
  a) Find the least square regression line for the given data points.
  b) Plot the given points and the regression line in the same rectangular system of axes.

- **Problem 2**

  a) Find the least square regression line for the following set of data

  {(-1 , 0),(0 , 2),(1 , 4),(2 , 5)}

  b) Plot the given points and the regression line in the same rectangular system of axes.

- **Problem 3**

  The values of y and their corresponding values of y are shown in the table below

x	0	1	2	3	4
y	2	3	5	4	6

  a) Find the least square regression line y = a x + b.
  b) Estimate the value of y when x = 10.

- **Problem 4**

  The sales of a company (in million dollars) for each year are shown in the table below.

x (year)	2005	2006	2007	2008	2009
y (sales)	12	19	29	37	45

  a) Find the least square regression line y = a x + b.
  b) Use the least squares regression line as a model to estimate the sales of the company in 2012.

Solutions to the Above Problems

1. a) Let us organize the data in a table.

x	**y**	**x y**	**x ^{2}**
-2	-1	2	4
1	1	1	1
3	2	6	9
$\sum x = 2$	$\sum y = 2$	$\sum xy = 9$	$\sum x^2 = 14$

We now use the above formula to calculate a and b as follows
$a = (n\sum x\, y - \sum x \sum y) / (n\sum x^2 - (\sum x)^2) = (3*9 - 2*2) / (3*14 - 2^2) = 23/38$
$b = (1/n)(\sum y - a \sum x) = (1/3)(2 - (23/38)*2) = 5/19$
b) We now graph the regression line given by y = a x + b and the given points.

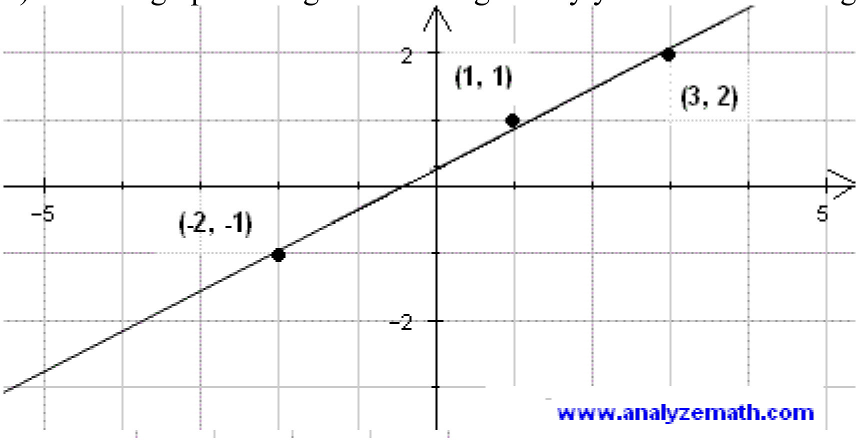


Figure 3. Graph of linear regression in problem 1.

2. a) We use a table as follows

x	**y**	**x y**	**x ^{2}**
-1	0	0	1
0	2	0	0
1	4	4	1
2	5	10	4
$\sum x = 2$	$\sum y = 11$	$\sum x\, y = 14$	$\sum x^2 = 6$

We now use the above formula to calculate a and b as follows
$a = (n\sum x\, y - \sum x \sum y) / (n\sum x^2 - (\sum x)^2) = (4*14 - 2*11) / (4*6 - 2^2) = 17/10 = 1.7$
$b = (1/n)(\sum y - a \sum x) = (1/4)(11 - 1.7*2) = 1.9$

+ b and the given points.

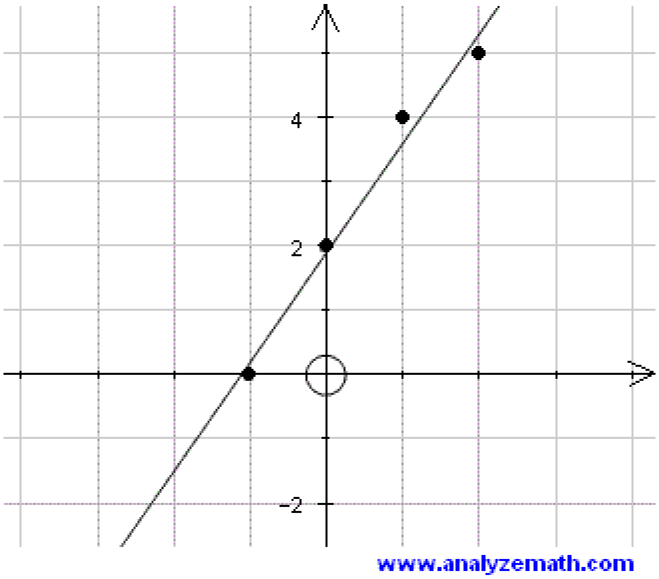


Figure 4. Graph of linear regression in problem 2.

3. a) We use a table to calculate a and b.

x	**y**	**x y**	**x** ^{2}
0	2	0	0
1	3	3	1
2	5	10	4
3	4	12	9
4	6	24	16
$\sum x = 10$	$\sum y = 20$	$\sum x\, y = 49$	$\sum x^2 = 30$

We now calculate a and b using the least square regression formulas for a and b.
$a = (n\sum x\, y - \sum x \sum y) / (n\sum x^2 - (\sum x)^2) = (5*49 - 10*20) / (5*30 - 10^2) = 0.9$
$b = (1/n)(\sum y - a \sum x) = (1/5)(20 - 0.9*10) = 2.2$
b) Now that we have the least square regression line $y = 0.9\,x + 2.2$, substitute x by 10 to

find the value of the corresponding y.
y = 0.9 * 10 + 2.2 = 11.2

a) We first change the variable x into t such that t = x - 2005 and therefore t represents the number of years after 2005. Using t instead of x makes the numbers smaller and therefore manageable. The table of values becomes.

t (years after 2005)	0	1	2	3	4
y (sales)	12	19	29	37	45

We now use the table to calculate a and b included in the least regression line formula.

t	**y**	**t y**	**t^2**
0	12	0	0
1	19	19	1
2	29	58	4
3	37	111	9
4	45	180	16
$\sum t = 10$	$\sum y = 142$	$\sum ty = 368$	$\sum t^2 = 30$

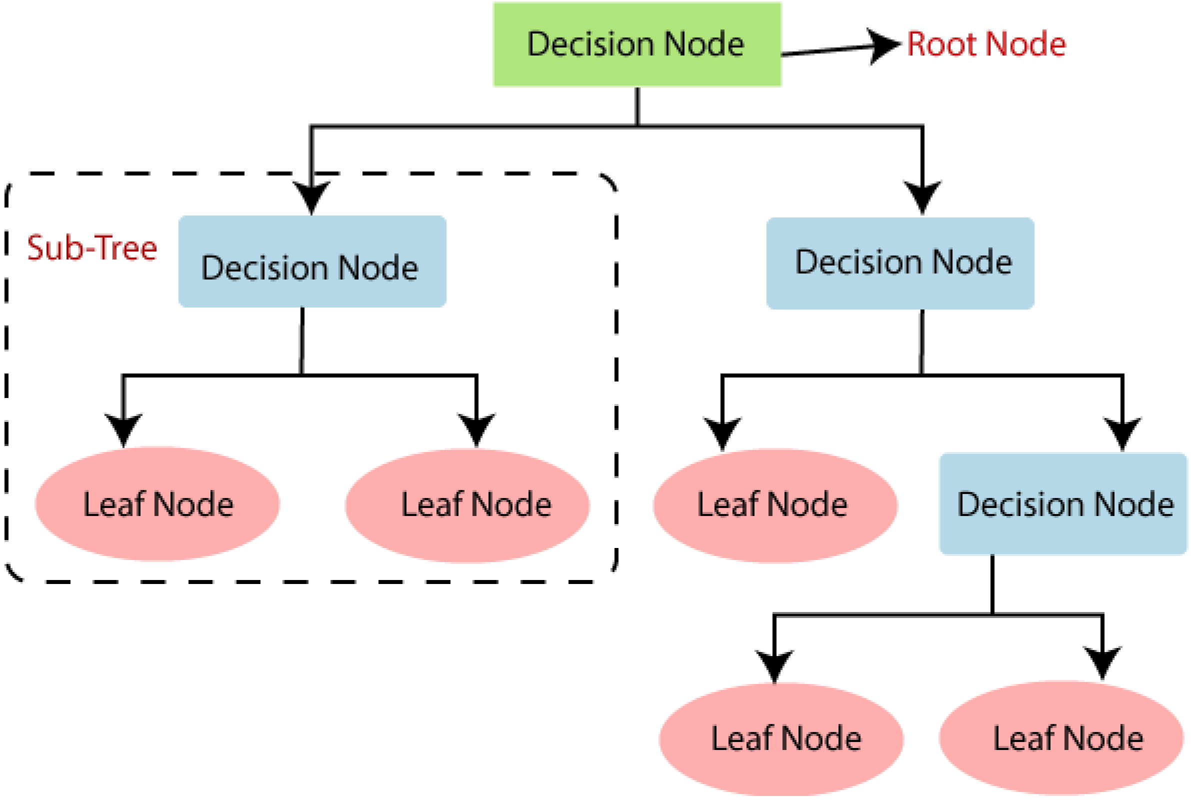
We now calculate a and b using the least square regression formulas for a and b.∑
$a = (n\sum t\, y - \sum t \sum y) / (n\sum t^2 - (\sum t)^2) = (5*368 - 10*142) / (5*30 - 10^2) = 8.4$
$b = (1/n)(\sum y - a \sum x) = (1/5)(142 - 8.4*10) = 11.6$
b) In 2012, t = 2012 - 2005 = 7
The estimated sales in 2012 are: y = 8.4 * 7 + 11.6 = 70.4 million dollars.

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- **Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.**
- **The logic behind the decision tree can be easily understood because it shows a tree-like structure.**

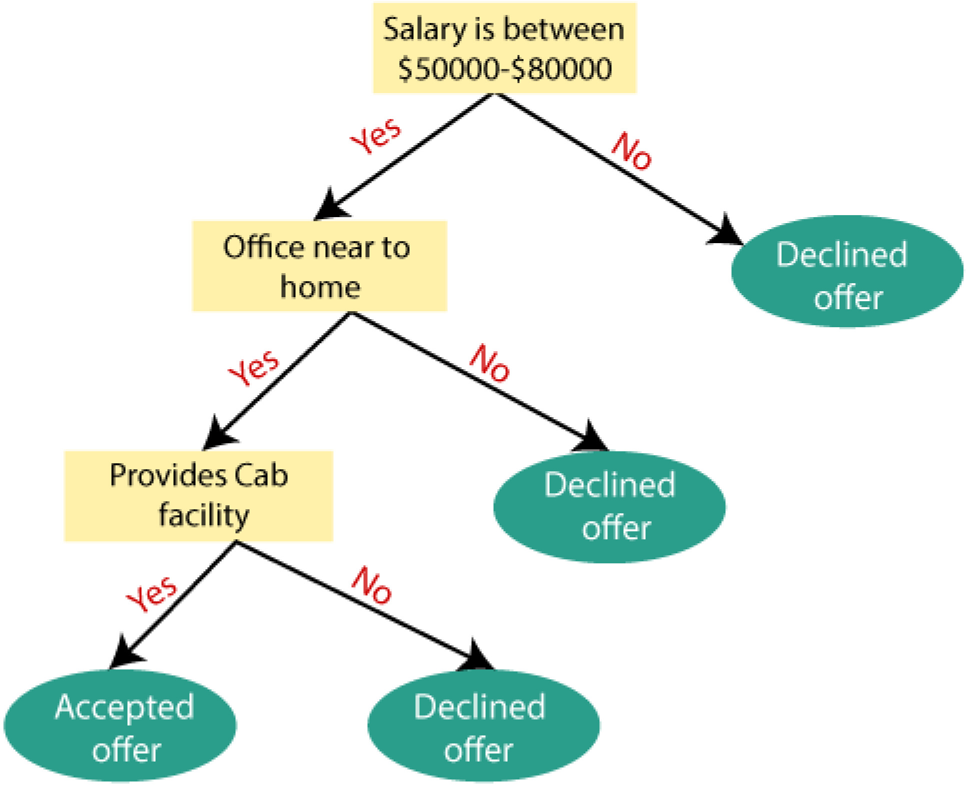
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Step-by-Step Working of Decision Tree Algorithm

Decision Tree Analysis is a generic predictive modeling tool with applications in various fields. Decision trees are generally built using an algorithmic approach that identifies multiple ways to segment a data set based on certain factors. It is one of the most extensively used and practical supervised learning algorithms. Decision Trees are a non-parametric supervised learning method that can be used for classification and regression applications. The goal is to build a model that predicts the value of a target variable using basic decision rules derived from data attributes.

The decision rules are typically written in the form of if-then-else expressions. The deeper the tree, the more complicated the rules and the more accurate the model.

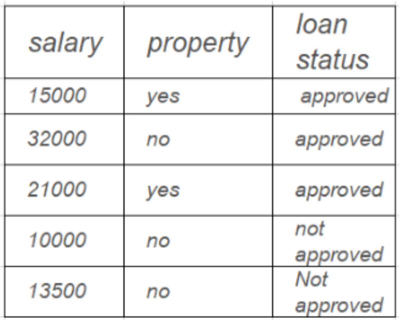
Learning Objective

This blog explores why decision trees are easy to interpret with their structure. We will see plenty of examples to understand the flow of the decision tree algorithm. Through hands-on demonstrations, we will also understand how to split the nodes of a decision tree. And we will finally implement it via python using a popular dataset.

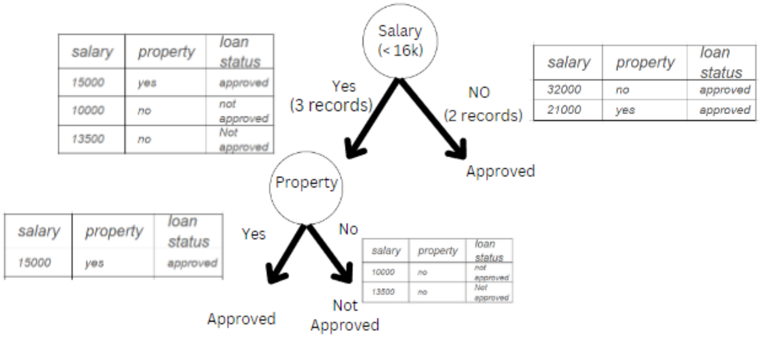
How Can We Create A Simple Decision Tree?

Decision trees are one of the most popular and accessible algorithms out there. It is very intuitive because it works exactly the way we think. For example, to decide about our career options or to buy a product or house. In simple lines, the daily decisions are identical to the decision tree model.

So, how was the structure built? And why it’s similar to the way we think. Let’s understand with an example below.



salary	property	loan status
15000	yes	approved
32000	no	approved
21000	yes	approved
10000	no	not approved
13500	no	Not approved



salary	property	loan status
15000	yes	approved
10000	no	not approved
13500	no	Not approved

salary	property	loan status
32000	no	approved
21000	yes	approved

salary	property	loan status
15000	yes	approved

salary	property	loan status
10000	no	not approved
13500	no	Not approved

Image 1

In the above example, we are trying to understand whether a person gets a loan based on salary and property. We can consider Y variable (loan approval) column. There are two input parameters: the X1 variable – salary(in rupees) and the X2 variable – property(land or house). We have built a small decision tree.

Condition 1: If the salary is less than Rs. 16000, we need to check whether they have property. If yes, give them the loan.

Condition 2: Give them a loan if the salary is more than Rs. 16000.

The example above is very straightforward to understand the structure. But before moving forward, we need to understand a few important questions.

- ***Question 1***-> What are terminologies used in a decision tree? We will understand with an example below.

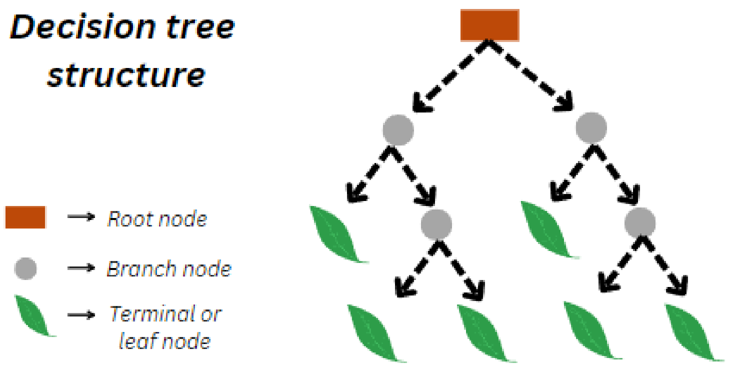


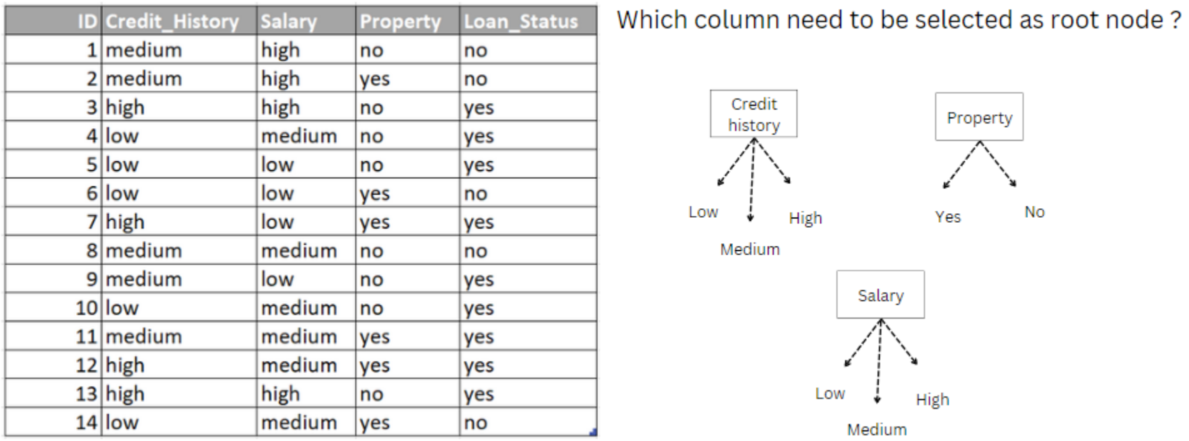
Image 2

- ***Question 2*** -> Why did we select the salary column first instead of the property column in Image 1? We considered the salary column as an example of building a tree. But, when we are working on the real-world dataset, we cannot choose the column randomly. Read the following section to know what process we use in real-time.

Splitting the Nodes in a Decision Tree

Now, Let’s consider the dataset below in Image 3 for a detailed understanding. Again we need to answer the question before building the decision tree model.

Which column must be selected as the root node in the dataset below?



Which column need to be selected as root node ?

ID	Credit_History	Salary	Property	Loan_Status
1	medium	high	no	no
2	medium	high	yes	no
3	high	high	no	yes
4	low	medium	no	yes
5	low	low	no	yes
6	low	low	yes	no
7	high	low	yes	yes
8	medium	medium	no	no
9	medium	low	no	yes
10	low	medium	no	yes
11	medium	medium	yes	yes
12	high	medium	yes	yes
13	high	high	no	yes
14	low	medium	yes	no

Credit history
Low
Medium
High
Property
Yes
No
Salary
Low
Medium
High

Image 3

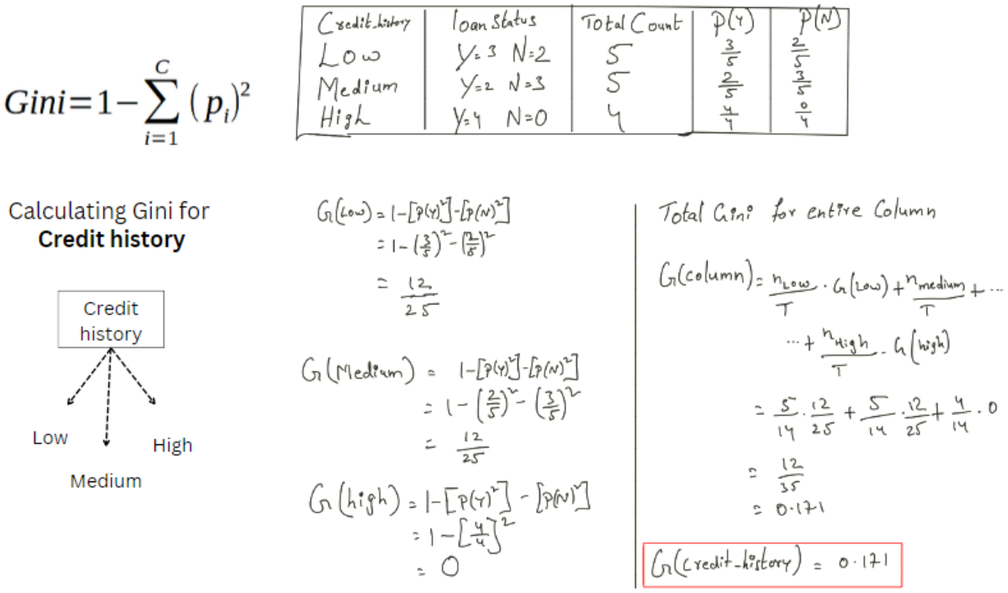
To answer the above question, we need to check how good each column is and What qualities it has to be a root node. To know which column we will be using:

1. Gini
2. Entropy and Information Gain

Let’s understand one by one with hands-on examples.

Gini Impurity in Decision Tree: How to Understand It?

First, We will calculate the Gini impurity for column 1 **credit history. Likewise**, we must calculate the Gini impurity for the other columns like **salary and property.** The value we will get is how impure an attribute is. So, the lesser the value lesser the impurity the value ranges between(0-1).



$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Credit_history	loan Status	Total Count	P(Y)	P(N)
Low	Y=3 N=2	5	$\frac{3}{5}$	$\frac{2}{5}$
Medium	Y=2 N=3	5	$\frac{2}{5}$	$\frac{3}{5}$
High	Y=4 N=0	4	$\frac{4}{4}$	$\frac{0}{4}$

Calculating Gini for **Credit history**

$$G(Low) = 1 - [P(Y)^2] - [P(N)^2]$$
$$= 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2$$
$$= \frac{12}{25}$$

$$G(Medium) = 1 - [P(Y)^2] - [P(N)^2]$$
$$= 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2$$
$$= \frac{12}{25}$$

$$G(high) = 1 - [P(Y)^2] - [P(N)^2]$$
$$= 1 - \left[\frac{4}{4}\right]^2$$
$$= 0$$

Total Gini for entire Column

$$G(column) = \frac{n_{Low}}{T} \cdot G(Low) + \frac{n_{medium}}{T} + \cdots$$
$$\cdots + \frac{n_{High}}{T} \cdot G(high)$$
$$= \frac{5}{14} \cdot \frac{12}{25} + \frac{5}{14} \cdot \frac{12}{25} + \frac{4}{14} \cdot 0$$
$$= \frac{12}{35}$$
$$= 0.171$$

$$\boxed{G(credit\_history) = 0.171}$$

Image 4

In the above image4, we got Gini for each class in credit history.

G(Low) = 12/25

G(medium) = 12/25

G(High) = 0

Then we calculated the Total Gini for the entire credit history column**.** Let’s understand the formula.

$$G(\text{column}) = \frac{n_{Low}}{T} \cdot G(\text{Low}) + \frac{n_{medium}}{T} + \frac{n_{High}}{T} \cdot G(\text{high})$$

Image 5

n = Total count of that class in the column credit history

T = Total count of instances. In our case, it is 14(we have a total of 14 rows in the Image3)

Example: nlow/T can be written as 5/14. Here we got 5 from the count of Credit_history(low) we see in Image 4 (3rd column).

$$= \frac{5}{14} \cdot \frac{12}{25} + \frac{5}{14} \cdot \frac{12}{25} + \frac{4}{14} \cdot 0$$

Image 6

Finally, We got the Impurity for the column **Credit history = 0.171.**

Now, We have to calculate the Gini for each Feature as we did for the credit_history feature above.

After calculating Gini for both features,

We will get Impurity for the column **Salary = 0.440** and **Property = 0.428.** For a detailed understanding, we can check the image below.

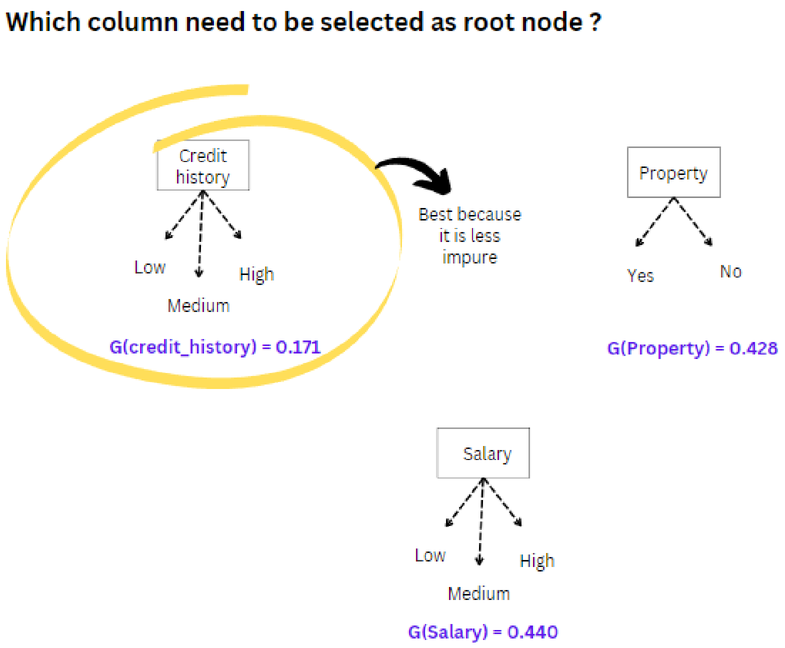

Which column need to be selected as root node ?
Credit history
Low
Medium
High
G(credit_history) = 0.171
Best because it is less impure
Property
Yes
No
G(Property) = 0.428
Salary
Low
Medium
High
G(Salary) = 0.440

Image 7

Similarly, we continue the process of selecting the branch node, and we can build a Decision Tree. We will now try a different method for carrying out the same process of building a decision tree below.

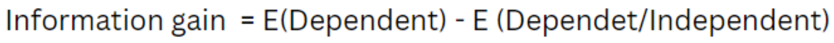
The Idea of Entropy and Information Gain

Entropy is also a measure of randomness or impurity. It is helpful while splitting the nodes further and making the right decision. The formula for entropy is:

$$\text{Entropy} = -p_{(Yes)} \log p_{(Yes)} - p_{(No)} \log p_{(No)}$$

Image 8

But with entropy, we will also be using Information gain. So, Information gain helps to understand how much information we get from each feature.

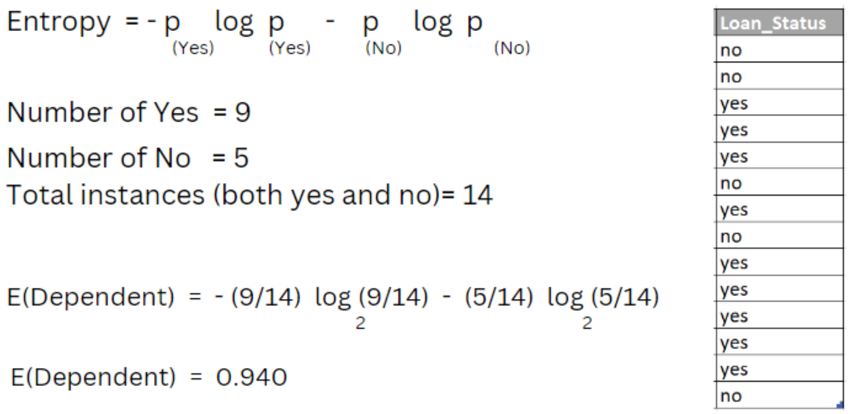


Information gain = E(Dependent) - E (Dependet/Independent)

Image 9

But Before going further, we need to understand why we need entropy and information gain and why we need to use them. Let’s consider the example we have used in Gini. In that, we have a **Credit history, Salary, and property.** Again we have to start from the beginning. What do I mean by beginning from scratch? We must start calculating Entropy and information gained from each attribute and select a root node. That means we have 2 popular ways of solving the problem 1. Gini, 2. Entropy and information gain. We have already learned how to build a decision tree using Gini. From here on, we will understand how to build a decision tree using the Entropy and information gain step by step.

Before calculating the entropy for input attributes, we need to calculate the entropy for the target or output variable. In our dataset, the output variable is **loan status.**



$$\text{Entropy} = -p_{(Yes)} \log p_{(Yes)} - p_{(No)} \log p_{(No)}$$

Number of Yes = 9

Number of No = 5

Total instances (both yes and no)= 14

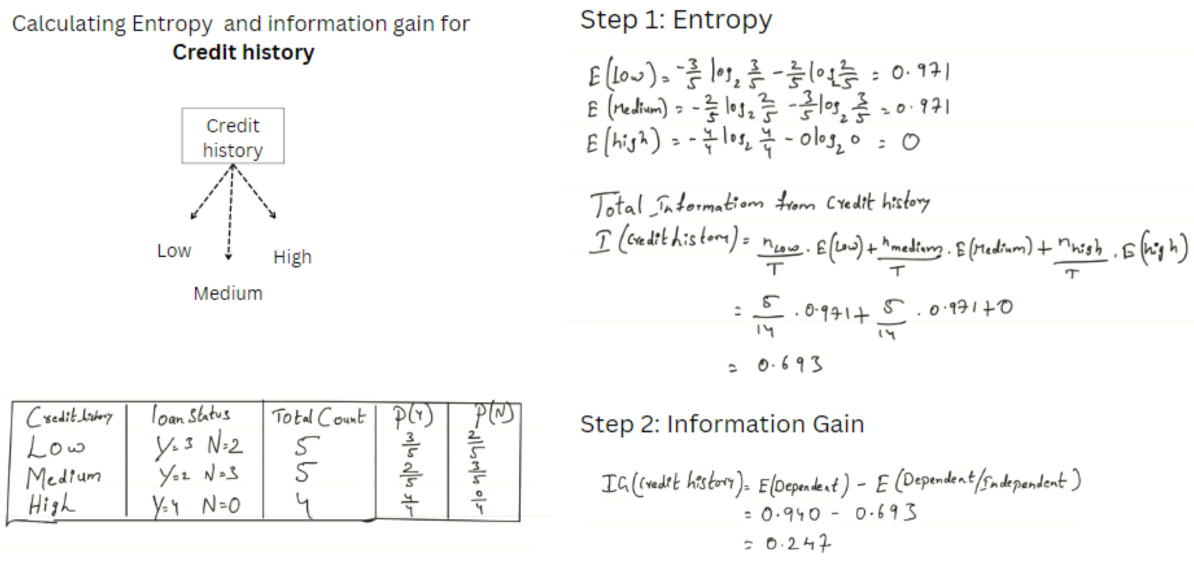
$$E(\text{Dependent}) = -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14)$$

$$E(\text{Dependent}) = 0.940$$

Loan_Status
no
no
yes
yes
yes
no
yes
no
yes
yes
yes
yes
yes
no

Image 10

Select the attribute out of 3 attributes as a root node in the data set we have seen in image 3. We need to calculate Information gain for all the 3 independent attributes.


Credit history
Low
Medium
High

Calculating Entropy and information gain for **Credit history**

Credit_history	loan Status	Total Count	P(Y)	P(N)
Low	Y=3 N=2	5	$\frac{3}{5}$	$\frac{2}{5}$
Medium	Y=2 N=3	5	$\frac{2}{5}$	$\frac{3}{5}$
High	Y=4 N=0	4	$\frac{4}{4}$	$\frac{0}{4}$

Step 1: Entropy

$$E(\text{Low}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$
$$E(\text{Medium}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$
$$E(\text{high}) = -\frac{4}{4}\log_2\frac{4}{4} - 0\log_2 0 = 0$$

Total Information from Credit history

$$I(\text{Credit history}) = \frac{n_{\text{Low}}}{T}\cdot E(\text{Low}) + \frac{n_{\text{medium}}}{T}\cdot E(\text{Medium}) + \frac{n_{\text{high}}}{T}\cdot E(\text{high})$$
$$= \frac{5}{14}\cdot 0.971 + \frac{5}{14}\cdot 0.971 + 0$$
$$= 0.693$$

Step 2: Information Gain

$$IG(\text{credit history}) = E(\text{Dependent}) - E(\text{Dependent}/\text{Independent})$$
$$= 0.940 - 0.693$$
$$= 0.247$$

Image 11

In the above image 11, we have calculated Entropy for the column Credit history. We can divide the calculations into two steps to understand the math above.

Step1: Entropy

We have calculated the entropy for each class in the **credit _history** column.

We got,

- E(Low) = 0.971
- E(Medium) = 0.971
- E(High) = 0

Then calculate the total information for the whole column **Credit history.**

At the end of step 1, we have done the same math as we have done in the Gini Impurity section above. We can refer to image 5 for a better understanding. Then we got a value.

I(Credit History) = 0.693

Step 2: Information Gain

This is the easiest step; we must subtract the I(independent) from E(Dependent). You can find the formula in images 9 and image 10.

IG(Credit History) = 0.247

We have to calculate the Entropy and information Gain for the remaining features as we did for the credit history feature above in image 11.

After repeating the same process for the other feature, we will get

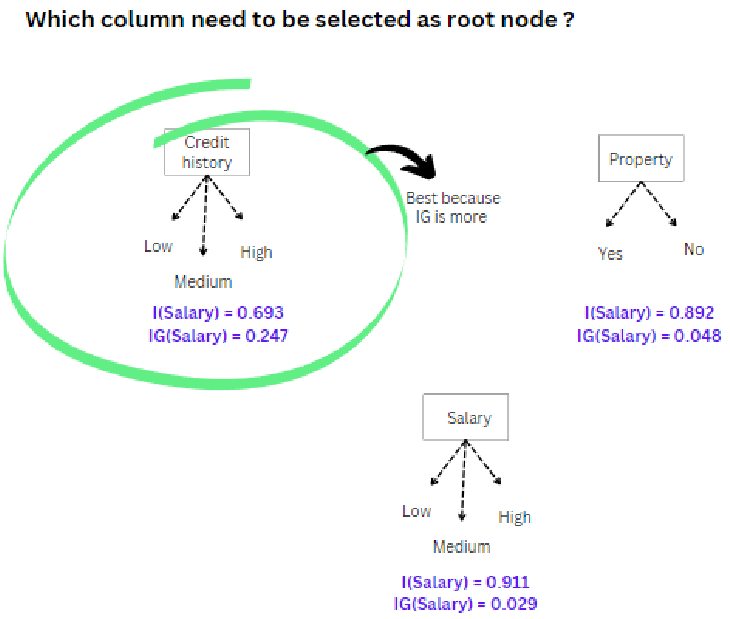


Image 12

The above image shows that credit history is the best attribute as a root node because the Information gain is more.

Similarly, we continue the process of selecting the branch node, and we can build a complete decision tree.

Advantages of the Decision Tree:

It is simple to understand as it follows the same process which a human follow while making any decision in real-life.

It can be very useful for solving decision-related problems.

It helps to think about all the possible outcomes for a problem.

There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree:

The decision tree contains lots of layers, which makes it complex.

It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

For more class labels, the computational complexity of the decision tree may increase.

What are appropriate problems for Decision tree learning?

Although a variety of decision tree learning methods have been developed with somewhat differing capabilities and requirements, decision tree learning is generally best suited to problems with the following characteristics:

1. Instances are represented by attribute-value pairs:

In the world of decision tree learning, we commonly use attribute-value pairs to represent instances. An instance is defined by a predetermined group of attributes, such as temperature, and its corresponding value, such as hot. Ideally, we want each attribute to have a finite set of distinct values, like hot, mild, or cold. This makes it easy to construct decision trees. However, more advanced versions of the algorithm can accommodate attributes with continuous numerical values, such as representing temperature with a numerical scale.

2. The target function has discrete output values:

The marked objective has distinct outcomes. The decision tree method is ordinarily employed for categorizing Boolean examples, such as yes or no. Decision tree approaches can be readily expanded for acquiring functions with beyond dual conceivable outcome values. A more substantial expansion lets us gain knowledge about aimed objectives with numeric outputs, although the practice of decision trees in this framework is comparatively rare.

3. Disjunctive descriptions may be required:

Decision trees naturally represent disjunctive expressions.

4.The training data may contain errors:

“Techniques of decision tree learning demonstrate high resilience towards discrepancies, including inconsistencies in categorization of sample cases and discrepancies in the feature details that characterize these cases.”

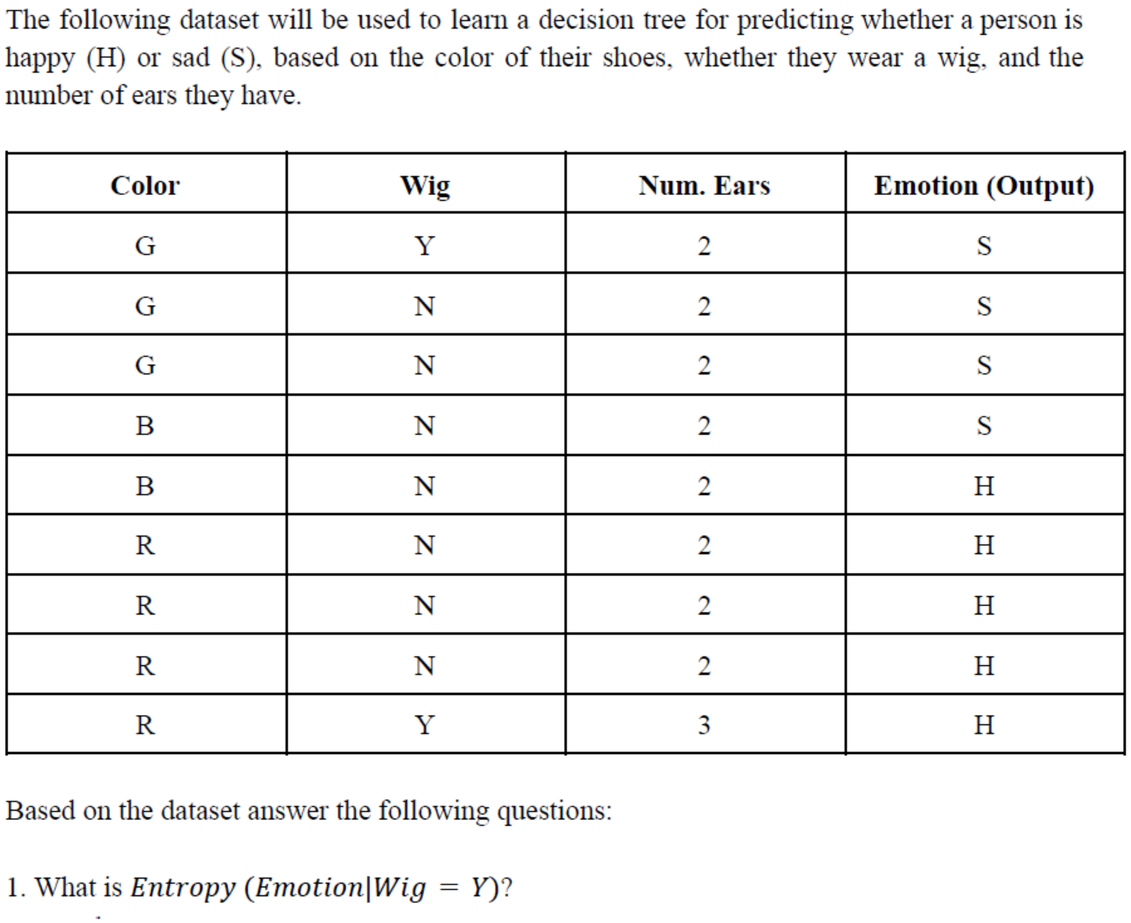
5. The training data may contain missing attribute values:

In certain cases, the input information designed for training might have absent characteristics. Employing decision tree approaches can still be possible despite experiencing unknown features in some training samples. For instance, when considering the level of humidity throughout the day, this information may only be accessible for a specific set of training specimens.

Practical issues in learning decision trees include:

- Determining how deeply to grow the decision tree,
- Handling continuous attributes,

- Choosing an appropriate attribute selection measure,
- Handling training data with missing attribute values,
- Handling attributes with differing costs, and
- Improving computational efficiency.

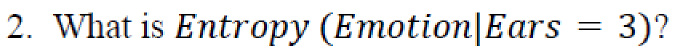


The following dataset will be used to learn a decision tree for predicting whether a person is happy (H) or sad (S), based on the color of their shoes, whether they wear a wig, and the number of ears they have.

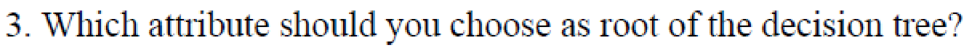
Color	Wig	Num. Ears	Emotion (Output)
G	Y	2	S
G	N	2	S
G	N	2	S
B	N	2	S
B	N	2	H
R	N	2	H
R	N	2	H
R	N	2	H
R	Y	3	H

Based on the dataset answer the following questions:

1. What is $Entropy\ (Emotion|Wig\ =\ Y)$?



2. What is $Entropy\ (Emotion|Ears\ =\ 3)$?



3. Which attribute should you choose as root of the decision tree?

To calculate the entropy of the target variable (*Emotion*) given the condition *Wig = Y*, we need to compute the distribution of emotions within that subset of the dataset.

Subset of the dataset where *Wig = Y*:

Color	Wig	Num. Ears	Emotion (Output)
G	Y	2	S
R	Y	3	H

Within this subset, we have 1 instance of "S" (sad) and 1 instance of "H" (happy). Therefore, the distribution of emotions is equal, with a count of 1 for each class.

To calculate the entropy, we can use the formula: $\text{Entropy}(X) = - \Sigma\, P(x) \log_2 P(x)$

$\mathit{Entropy}\ (\mathit{Emotion} \mid \mathit{Wig} = Y) = - P(S) \log_2 P(S) - P(H) \log_2 P(H)$

Since $P(S) = P(H) = 0.5$ (both classes have equal counts), we can substitute these values into the entropy formula:

$\mathit{Entropy}\ (\mathit{Emotion} \mid \mathit{Wig} = Y) = - (0.5) \log_2 (0.5) - (0.5) \log_2 (0.5)$

$$= - (0.5)\,(-1) - (0.5)\,(-1) = 1$$

Therefore, $\mathit{Entropy}\ (\mathit{Emotion} | \mathit{Wig}\ = Y) = 1$

Explanation:

To calculate the entropy of the target variable (Emotion) given the condition *Ears = 3*, we need to compute the distribution of emotions within that subset of the dataset.

Subset of the dataset where *Ears = 3*:

Color	Wig	Num. Ears	Emotion (Output)
R	Y	3	H

Within this subset, we have 1 instance of "H" (happy) and 0 instances of "S" (sad).

To calculate the entropy, we can use the formula:
Entropy(X) = - Σ P(x) $\log_2$ P(x)

Entropy (Emotion | *Ears=3)* = - P(S) $\log_2$ P(S) - P(H) $\log_2$ P(H)

Since P(S) = 0 and P(H) = 1 (since there are no instances of "S" and 1 instance of "H"), we can substitute these values into the entropy formula:

Entropy (Emotion | *Ears=3)* = - 0 $\log_2$ 0 - 1 $\log_2$ 1 = 0 - 0 = 0

Therefore, $Entropy\ (Emotion|Ears\ =\ 3)$ = 0.

3.

Information Gain (Color):

To calculate the information gain for the Color attribute, we need to compute the entropy of the Emotion variable before and after the split based on different colors.
Entropy (Emotion) = - (4/9) $\log_2$ (4/9) - (5/9) $\log_2$ (5/9) ≈ 0.991

After the split based on Color, we have the following subsets:

Subset for Color = Green:
Entropy (Emotion | Color = Green) = 0 (as all instances are of the same class, "S")

Subset for Color = Blue:
Entropy (Emotion | Color = Blue) = -1/2 $\log_2$ (1/2) - 1/2 $\log_2$(1/2) = 1

Subset for Color = Red:
Entropy (Emotion | Color = Red) = 0 (as all instances are of the same class, "H")

Information Gain (Color) = Entropy (Emotion) - [(3/9) * 0 + (2/9) * 1 + (4/9) * 0] ≈ 0.7687

Information Gain (Wig):

To calculate the information gain for the Wig attribute, we need to compute the entropy of the Emotion variable before and after the split based on different values of Wig.
Entropy (Emotion) = - (4/9) $\log_2$ (4/9) - (5/9) $\log_2$ (5/9) ≈ 0.991

After the split based on Wig, we have the following subsets:

Subset for Wig = Yes:
Entropy (Emotion | Wig = Yes) = -1/2 $\log_2$ (1/2) - 1/2 $\log_2$ (1/2) =1

Subset for Wig = No:
Entropy (Emotion | Wig = No) = - (4/7) $\log_2$ (4/7) - (3/7) $\log_2$ (3/7) ≈ 0.985

Information Gain (Wig) = Entropy (Emotion) - [(2/9) * 1 + (7/9) * 0.985] ≈ 0.002

Information Gain (Num. Ears):

To calculate the information gain for the Num. Ears attribute, we need to compute the entropy of the Emotion variable before and after the split based on different values of Num. Ears.
Entropy (Emotion) = - (4/9) $\log_2$ (4/9) - (5/9) $\log_2$ (5/9) ≈ 0.991

After the split based on Num. Ears, we have the following subsets:

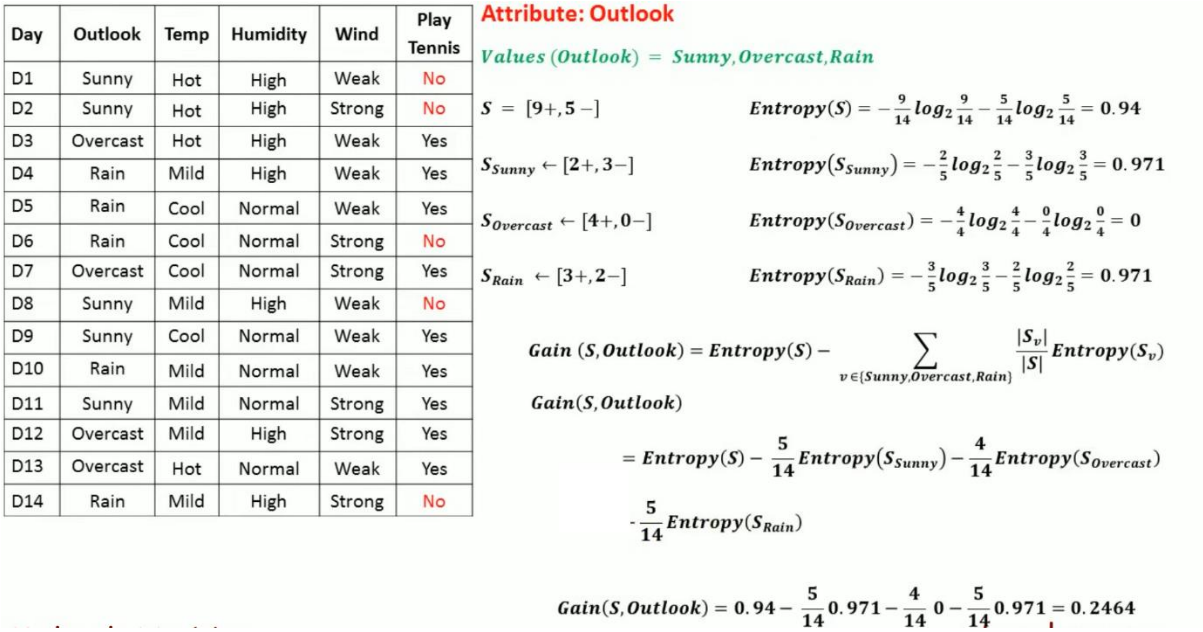
Subset for Num. Ears = 2:
Entropy (Emotion | Num. Ears = 2) = - (4/8) $\log_2$ (4/8) - (4/8) $\log_2$ (4/8) ≈ 1

Subset for Num. Ears = 3:
Entropy (Emotion | Num. Ears = 3) = 0 (as all instances are of the same class, "H")

Information Gain (Num. Ears) = Entropy (Emotion) - [(8/9) * 1 + (1/9) * 0] ≈ 0.102

To determine the attribute to choose as the root of the decision tree, we need to consider the concept of information gain. Information gain measures the reduction in entropy or impurity achieved by splitting the data based on a specific attribute.

We can calculate the information gain for each attribute by comparing the entropy before and after the split. The attribute with the highest information gain will be chosen as the root of the decision tree.



Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute: Outlook

$Values\ (Outlook) = Sunny, Overcast, Rain$

$S = [9+, 5-]$ $\qquad Entropy(S) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.94$

$S_{Sunny} \leftarrow [2+, 3-]$ $\qquad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$

$S_{Overcast} \leftarrow [4+, 0-]$ $\qquad Entropy(S_{Overcast}) = -\frac{4}{4}log_2\frac{4}{4} - \frac{0}{4}log_2\frac{0}{4} = 0$

$S_{Rain} \leftarrow [3+, 2-]$ $\qquad Entropy(S_{Rain}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$

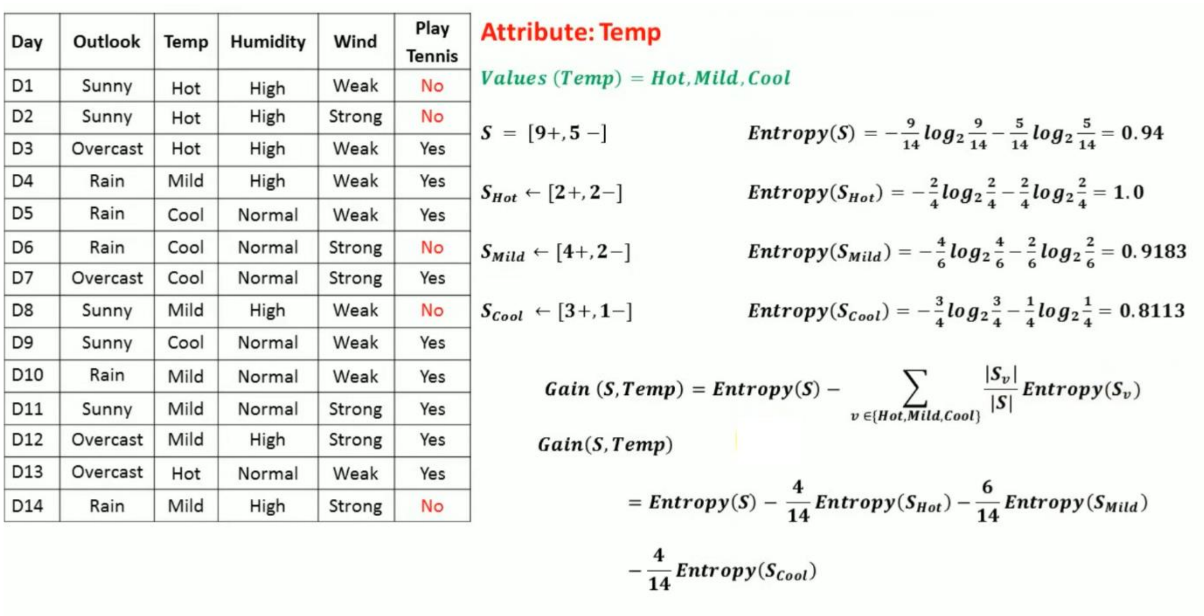
$$Gain\ (S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Outlook)$$

$$= Entropy(S) - \frac{5}{14}Entropy(S_{Sunny}) - \frac{4}{14}Entropy(S_{Overcast})$$

$$-\frac{5}{14}Entropy(S_{Rain})$$

$$Gain(S, Outlook) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}\,0 - \frac{5}{14}0.971 = 0.2464$$



Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute: Temp

$\boldsymbol{Values\ (Temp) = Hot, Mild, Cool}$

$S = [9+, 5-]$ $\qquad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{Hot} \leftarrow [2+, 2-]$ $\qquad Entropy(S_{Hot}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.0$

$S_{Mild} \leftarrow [4+, 2-]$ $\qquad Entropy(S_{Mild}) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.9183$

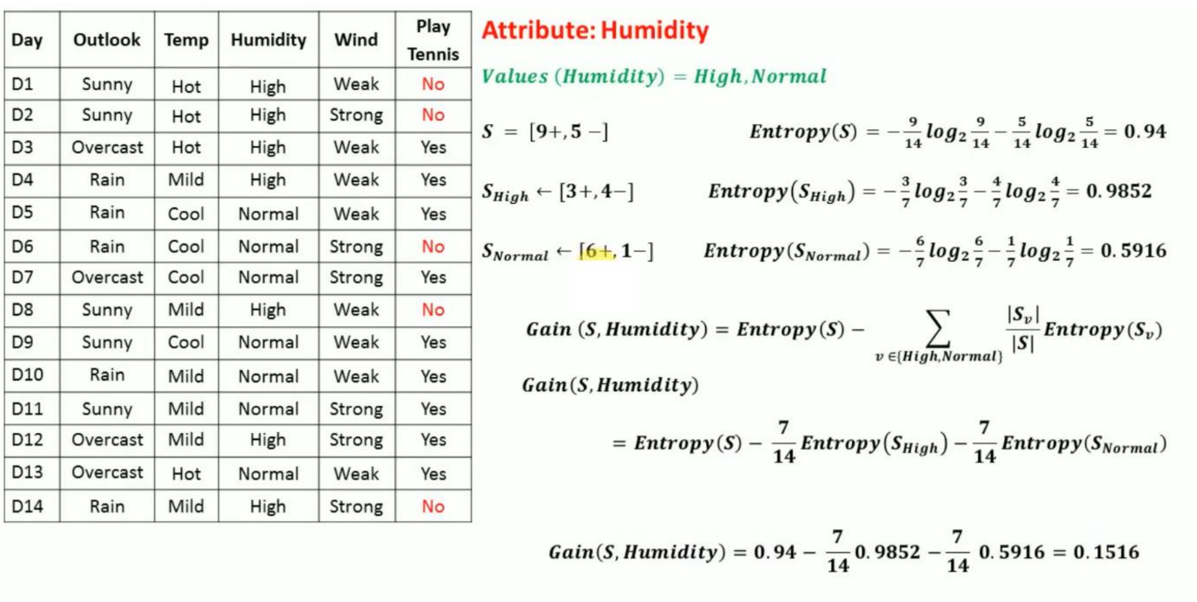
$S_{Cool} \leftarrow [3+, 1-]$ $\qquad Entropy(S_{Cool}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8113$

$$Gain\ (S, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Gain(S, Temp)$

$$= Entropy(S) - \frac{4}{14} Entropy(S_{Hot}) - \frac{6}{14} Entropy(S_{Mild})$$

$$- \frac{4}{14} Entropy(S_{Cool})$$



Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute: Humidity

$\boldsymbol{Values\ (Humidity) = High, Normal}$

$S = [9+, 5-]$ $\qquad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{High} \leftarrow [3+, 4-]$ $\qquad Entropy(S_{High}) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.9852$

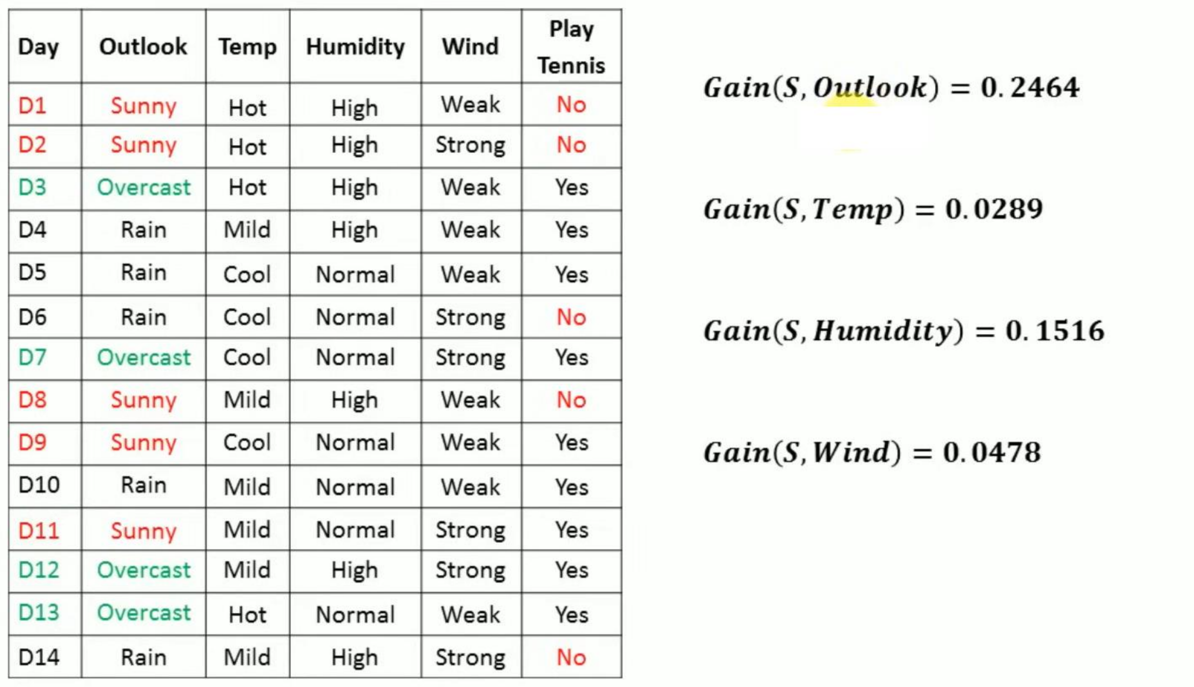
$S_{Normal} \leftarrow [6+, 1-]$ $\qquad Entropy(S_{Normal}) = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.5916$

$$Gain\ (S, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Gain(S, Humidity)$

$$= Entropy(S) - \frac{7}{14} Entropy(S_{High}) - \frac{7}{14} Entropy(S_{Normal})$$

$$Gain(S, Humidity) = 0.94 - \frac{7}{14} 0.9852 - \frac{7}{14}\ 0.5916 = 0.1516$$



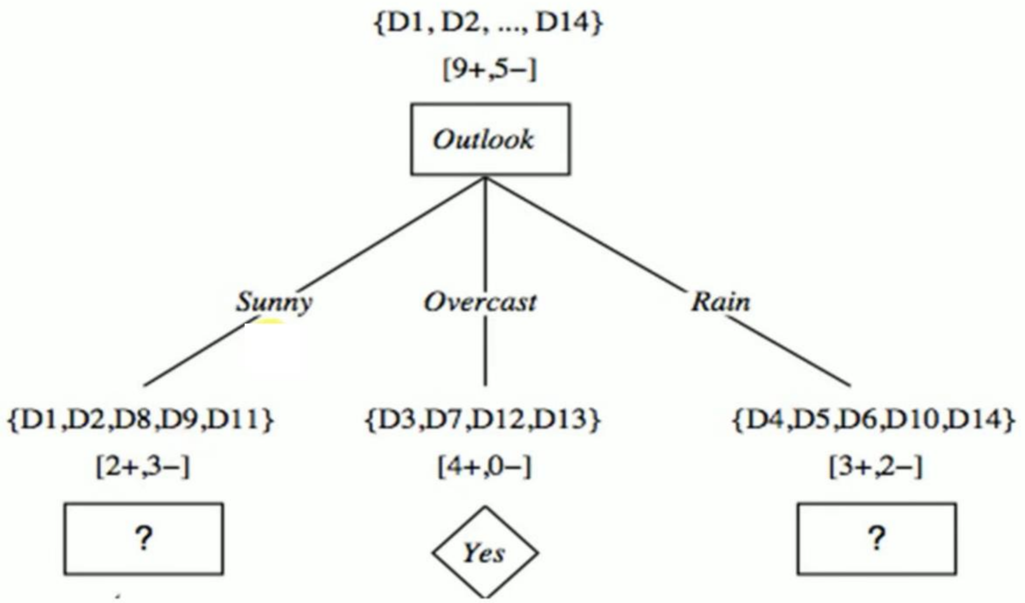
Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

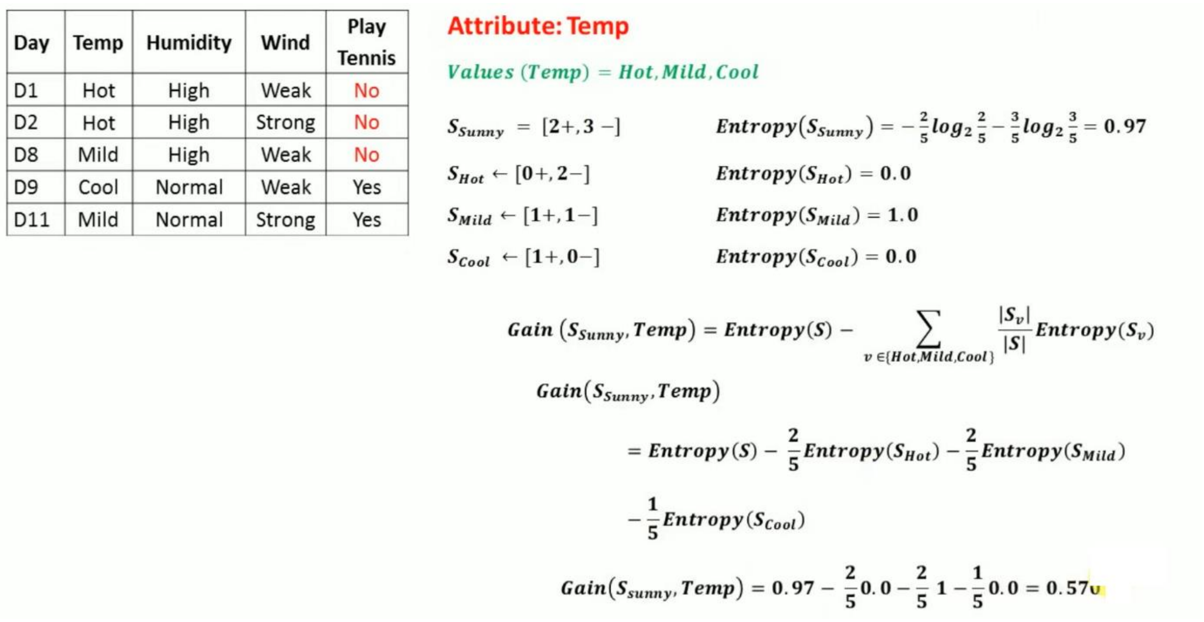
$$\boldsymbol{Gain(S, Outlook) = 0.2464}$$

$$\boldsymbol{Gain(S, Temp) = 0.0289}$$

$$\boldsymbol{Gain(S, Humidity) = 0.1516}$$

$$\boldsymbol{Gain(S, Wind) = 0.0478}$$





Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Attribute: Temp

$Values\ (Temp) = Hot, Mild, Cool$

$S_{Sunny} = [2+, 3-]$ $\qquad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 2-]$ $\qquad Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [1+, 1-]$ $\qquad Entropy(S_{Mild}) = 1.0$

$S_{Cool} \leftarrow [1+, 0-]$ $\qquad Entropy(S_{Cool}) = 0.0$

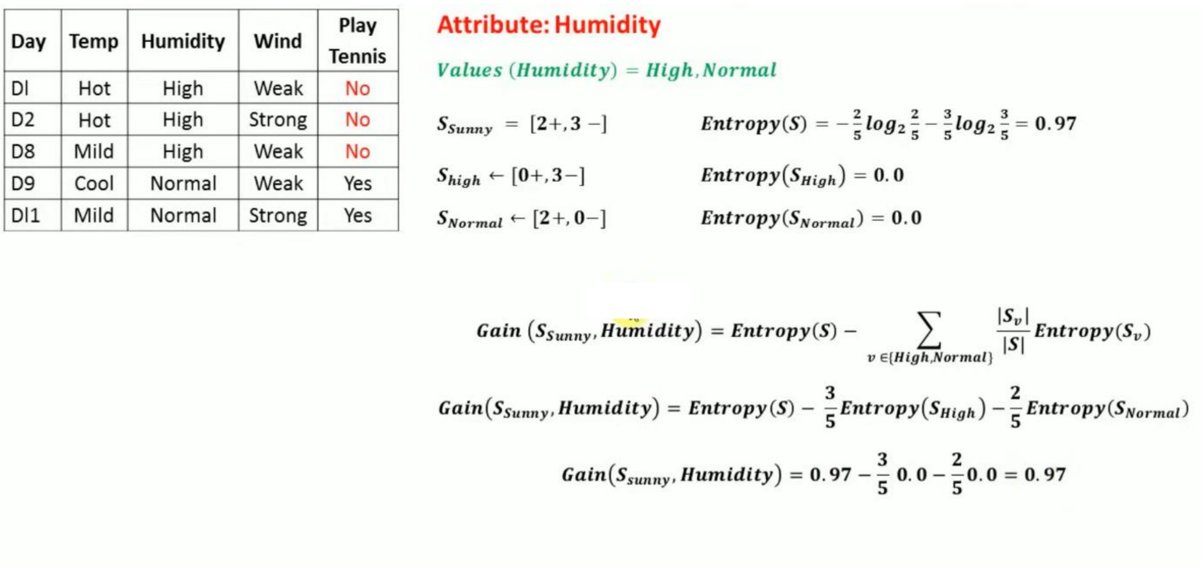
$$Gain\ (S_{Sunny}, Temp) = Entropy(S) - \sum_{v\ \in\{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Temp)$$

$$= Entropy(S) - \frac{2}{5}Entropy(S_{Hot}) - \frac{2}{5}Entropy(S_{Mild})$$

$$-\frac{1}{5}Entropy(S_{Cool})$$

$$Gain(S_{sunny}, Temp) = 0.97 - \frac{2}{5}0.0 - \frac{2}{5}1 - \frac{1}{5}0.0 = 0.570$$



Day	Temp	Humidity	Wind	Play Tennis
Dl	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
Dl1	Mild	Normal	Strong	Yes

Attribute: Humidity

$Values\ (Humidity) = High, Normal$

$S_{Sunny} = [2+, 3-]$ $\qquad Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

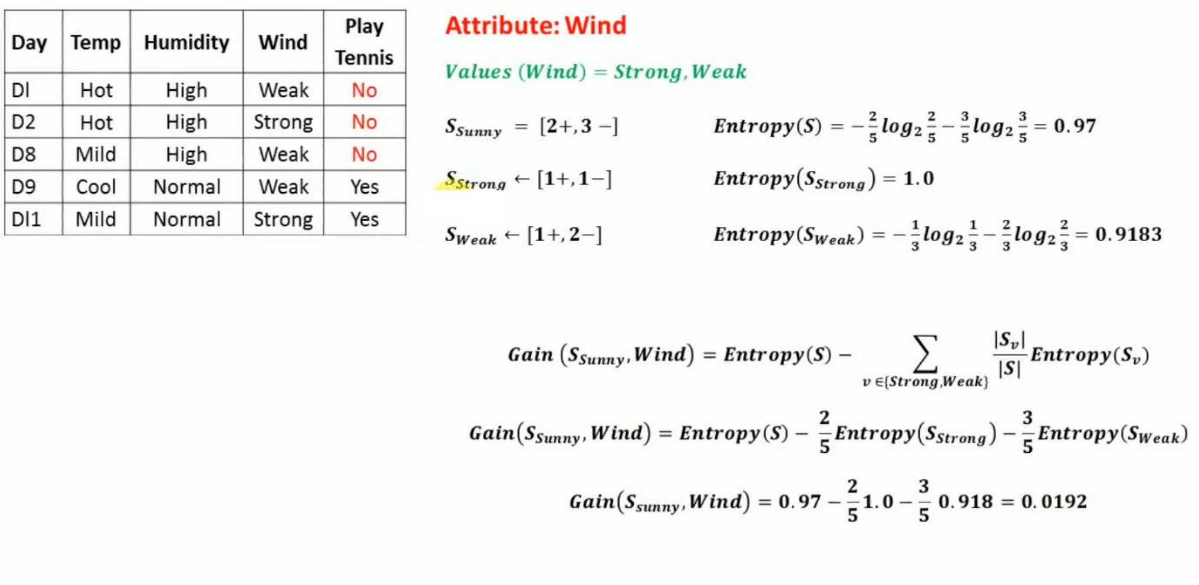
$S_{high} \leftarrow [0+, 3-]$ $\qquad Entropy(S_{High}) = 0.0$

$S_{Normal} \leftarrow [2+, 0-]$ $\qquad Entropy(S_{Normal}) = 0.0$

$$Gain\ (S_{Sunny}, Humidity) = Entropy(S) - \sum_{v\ \in\{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Humidity) = Entropy(S) - \frac{3}{5}Entropy(S_{High}) - \frac{2}{5}Entropy(S_{Normal})$$

$$Gain(S_{sunny}, Humidity) = 0.97 - \frac{3}{5}\ 0.0 - \frac{2}{5}0.0 = 0.97$$



Day	Temp	Humidity	Wind	Play Tennis
Dl	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
Dl1	Mild	Normal	Strong	Yes

Attribute: Wind

$Values\ (Wind) = Strong, Weak$

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

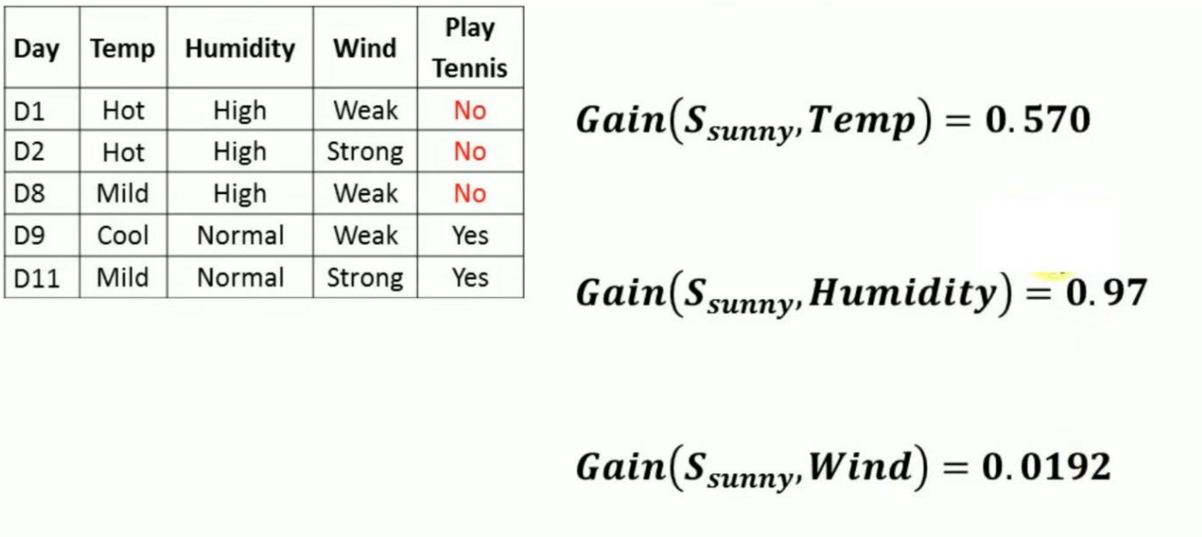
$S_{Strong} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Strong}) = 1.0$

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

$$Gain\ (S_{Sunny}, Wind) = Entropy(S) - \sum_{v\in\{Strong,Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Wind) = Entropy(S) - \frac{2}{5}Entropy(S_{Strong}) - \frac{3}{5}Entropy(S_{Weak})$$

$$Gain(S_{sunny}, Wind) = 0.97 - \frac{2}{5}1.0 - \frac{3}{5}0.918 = 0.0192$$

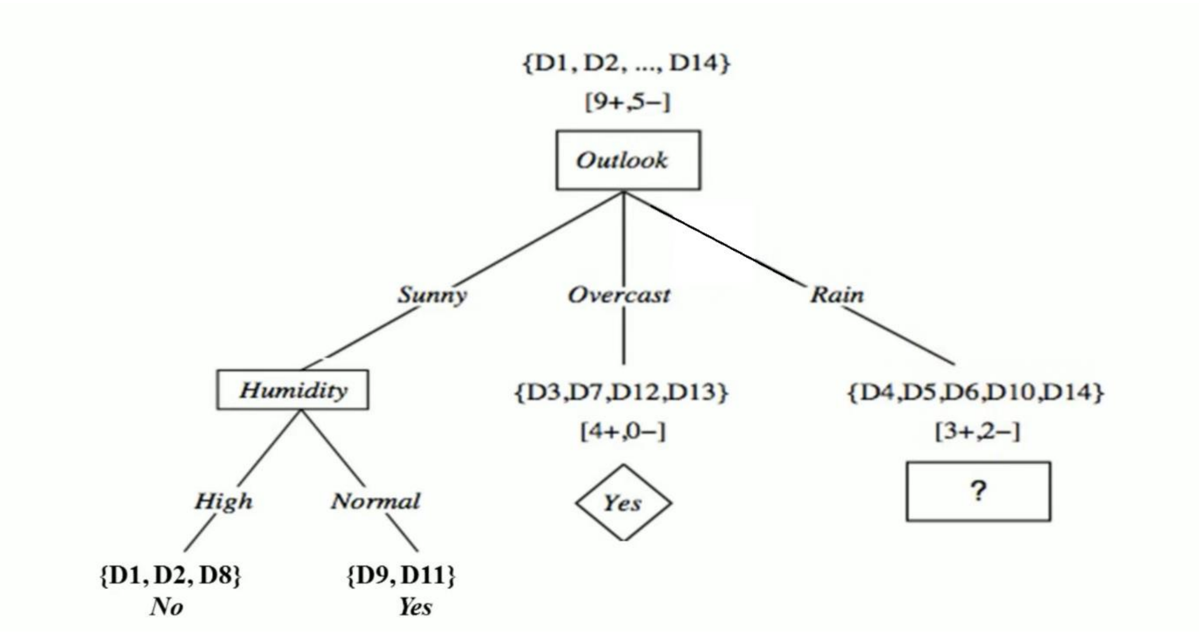


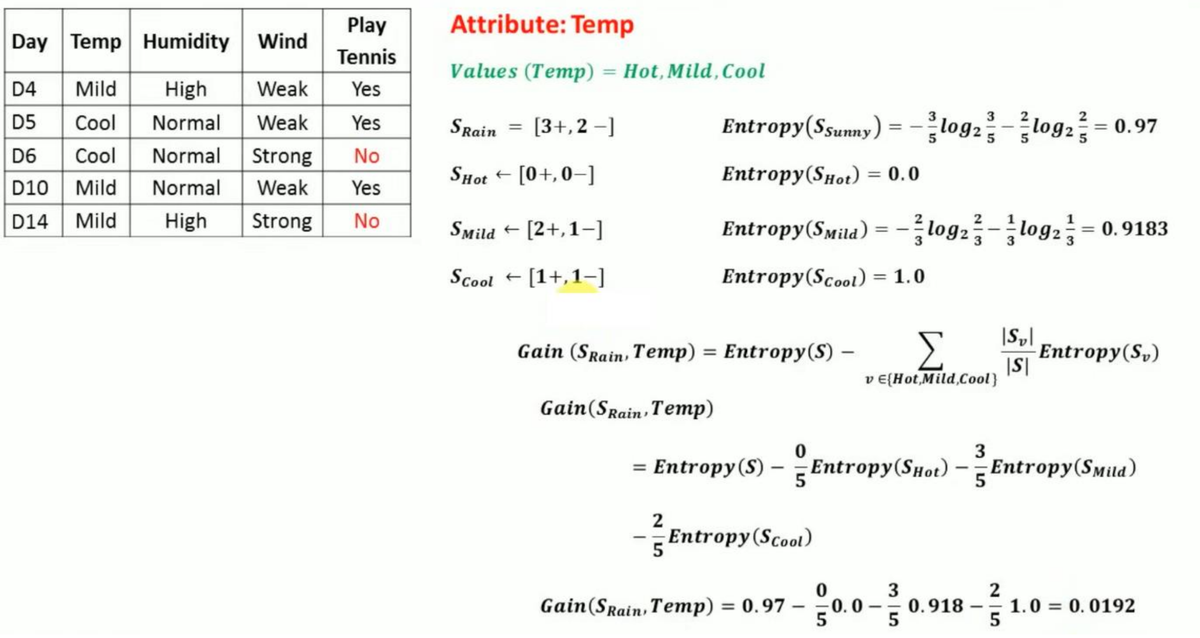
Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$$Gain(S_{sunny}, Temp) = 0.570$$

$$Gain(S_{sunny}, Humidity) = 0.97$$

$$Gain(S_{sunny}, Wind) = 0.0192$$





Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Attribute: Temp

$\boldsymbol{Values\ (Temp) = Hot, Mild, Cool}$

$S_{Rain} = [3+, 2\ -]$ $\qquad Entropy(S_{Sunny}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 0-]$ $\qquad Entropy(S_{Hot}) = 0.0$

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

$S_{Cool} \leftarrow [1+, 1-]$ $\qquad Entropy(S_{Cool}) = 1.0$

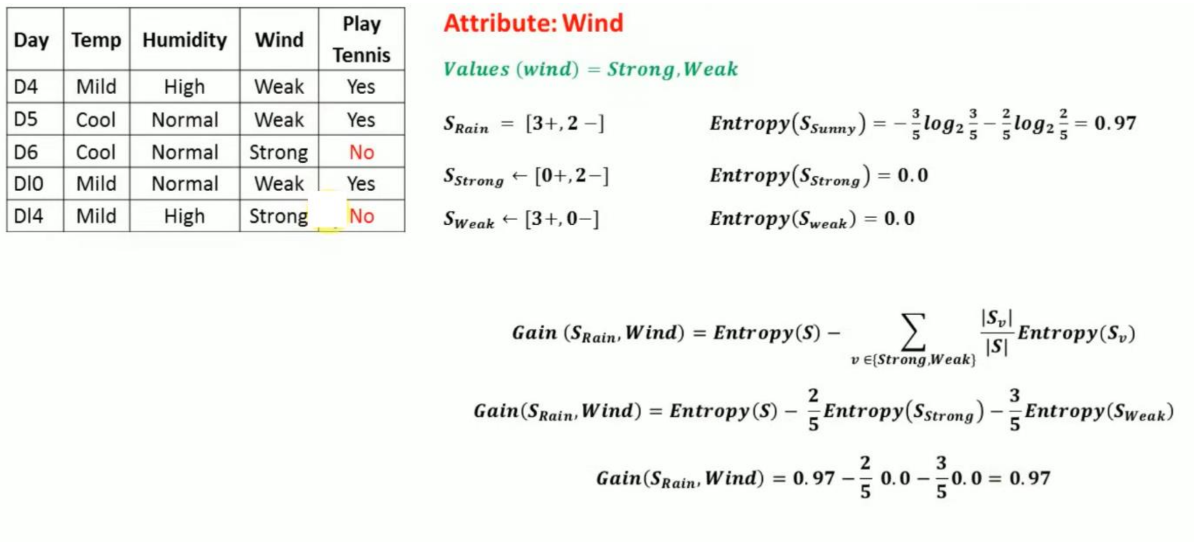
$$Gain\ (S_{Rain}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Temp)$$

$$= Entropy(S) - \frac{0}{5}Entropy(S_{Hot}) - \frac{3}{5}Entropy(S_{Mild})$$

$$-\frac{2}{5}Entropy(S_{Cool})$$

$$Gain(S_{Rain}, Temp) = 0.97 - \frac{0}{5}0.0 - \frac{3}{5}\,0.918 - \frac{2}{5}\,1.0 = 0.0192$$



Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
Dl0	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

Attribute: Wind

$Values\ (wind) = Strong, Weak$

$S_{Rain} = [3+, 2\ -]$ $\qquad Entropy(S_{Sunny}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.97$

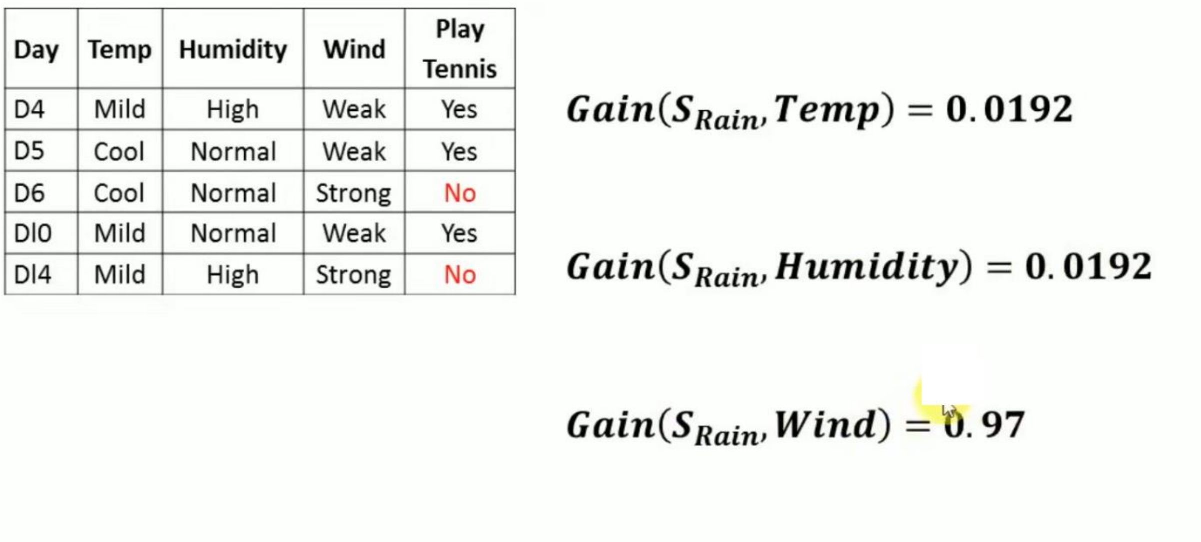
$S_{Strong} \leftarrow [0+, 2-]$ $\qquad Entropy(S_{Strong}) = 0.0$

$S_{Weak} \leftarrow [3+, 0-]$ $\qquad Entropy(S_{weak}) = 0.0$

$$Gain\ (S_{Rain}, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Rain}, Wind) = Entropy(S) - \frac{2}{5}Entropy(S_{Strong}) - \frac{3}{5}Entropy(S_{Weak})$$

$$Gain(S_{Rain}, Wind) = 0.97 - \frac{2}{5}\,0.0 - \frac{3}{5}0.0 = 0.97$$

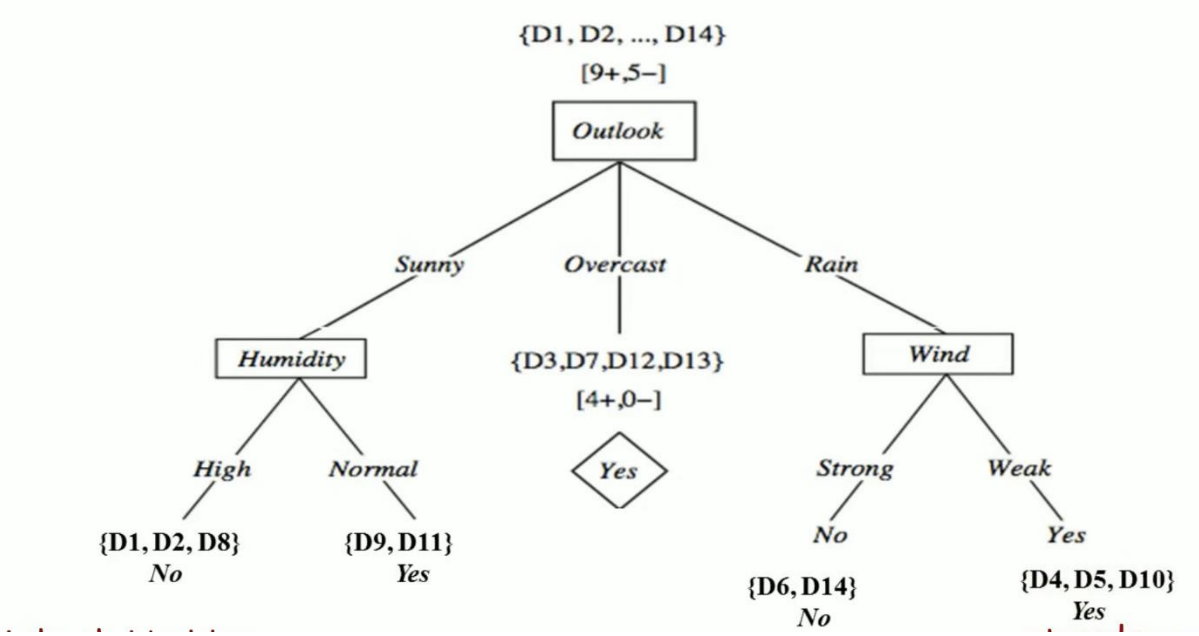


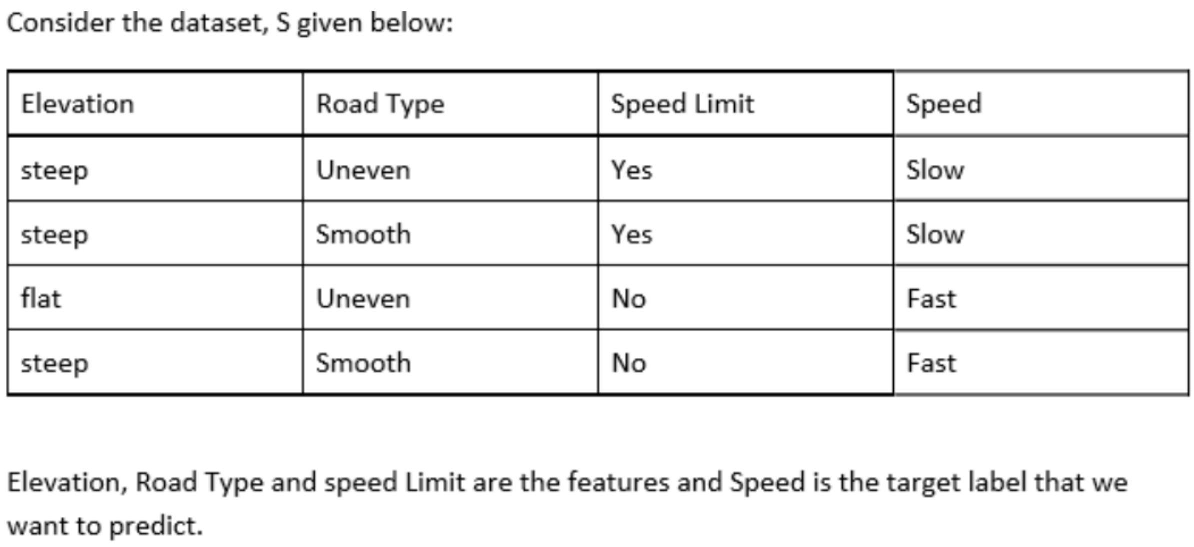
Day	Temp	Humidity	Wind	Play Tennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
Dl0	Mild	Normal	Weak	Yes
Dl4	Mild	High	Strong	No

$$Gain(S_{Rain}, Temp) = 0.0192$$

$$Gain(S_{Rain}, Humidity) = 0.0192$$

$$Gain(S_{Rain}, Wind) = 0.97$$

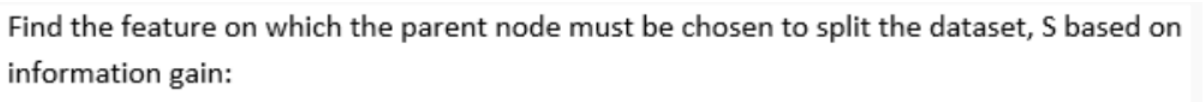




Consider the dataset, S given below:

Elevation	Road Type	Speed Limit	Speed
steep	Uneven	Yes	Slow
steep	Smooth	Yes	Slow
flat	Uneven	No	Fast
steep	Smooth	No	Fast

Elevation, Road Type and speed Limit are the features and Speed is the target label that we want to predict.



Find the feature on which the parent node must be chosen to split the dataset, S based on information gain:

Support Vector Machine (SVM) Algorithm

Introduction

SVM **is a powerful supervised algorithm that works best on** smaller datasets but on complex ones. Support Vector Machine, abbreviated as **SVM can be used for both regression and classification tasks, but generally, they work best in classification problems.**

What is a Support Vector Machine(SVM)?

It is a supervised machine learning problem where we try to find a hyperplane that best separates the two classes. **Note:** Don't get confused between SVM and logistic regression. Both the algorithms try to find the best hyperplane, but the main difference is logistic regression is a probabilistic approach whereas support vector machine is based on statistical approaches.

Now the question is which hyperplane does it select? There can be an infinite number of hyperplanes passing through a point and classifying the two classes perfectly. So, which one is the best?

Well, SVM does this by finding the maximum margin between the hyperplanes that means maximum distances between the two classes.

Logistic Regression vs Support Vector Machine (SVM)

Depending on the number of features you have you can either choose Logistic Regression or SVM.

SVM works best when the dataset is small and complex. It is usually advisable to first use logistic regression and see how does it performs, if it fails to give a good accuracy you can go for SVM without any kernel (will talk more about kernels in the later section). Logistic regression and SVM without any kernel have similar performance but depending on your features, one may be more efficient than the other.

Types of Support Vector Machine (SVM) Algorithms

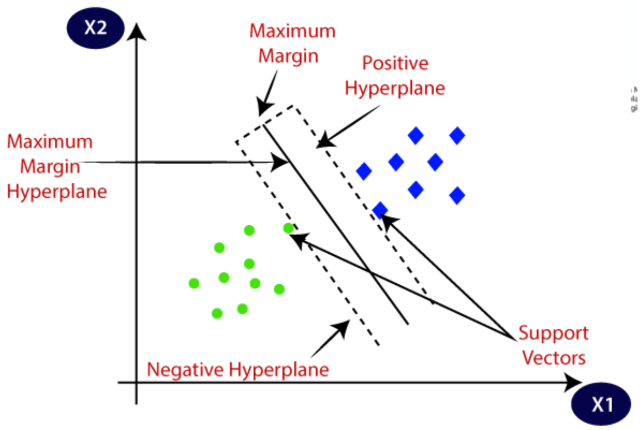
- **Linear SVM**: When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).
- **Non-Linear SVM**: When the data is not linearly separable then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a straight line (if 2D) then we use some advanced techniques like kernel tricks to classify them. In most real-world applications we do not find linearly separable datapoints hence we use kernel trick to solve them.

Important Terms

Now let's define two main terms which will be repeated again and again in this article:

- **Support Vectors:** These are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points.

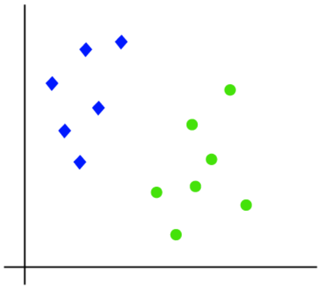
- **Margin:** it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. There are two types of margins **hard margin** and **soft margin.** I will talk more about these two in the later section.



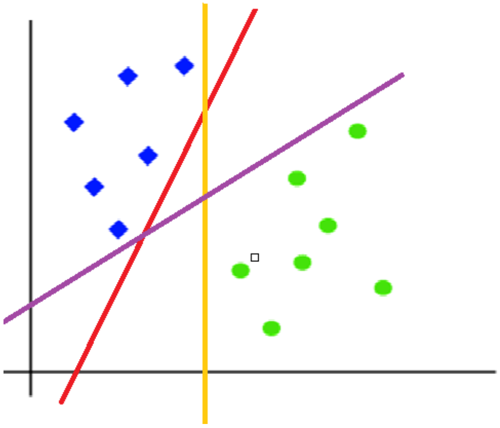
How Does Support Vector Machine Work?

SVM is defined such that it is defined in terms of the support vectors only, we don't have to worry about other observations since the margin is made using the points which are closest to the hyperplane (support vectors), whereas in logistic regression the classifier is defined over all the points. Hence SVM enjoys some natural speed-ups.

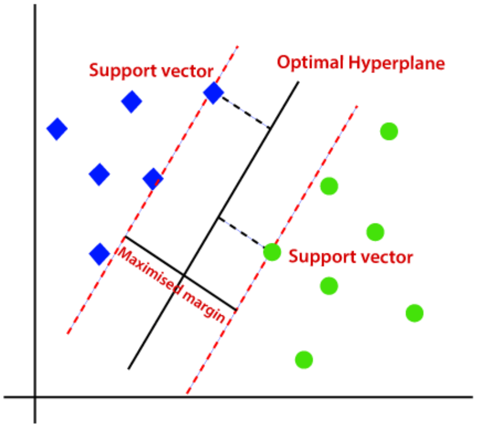
Let's understand the working of SVM using an example. Suppose we have a dataset that has two classes (green and blue). We want to classify that the new data point as either blue or green.



To classify these points, we can have many decision boundaries, but the question is which is the best and how do we find it? **NOTE:** Since we are plotting the data points in a 2-dimensional graph we call this decision boundary a **straight line** but if we have more dimensions, we call this decision boundary a **"hyperplane"**

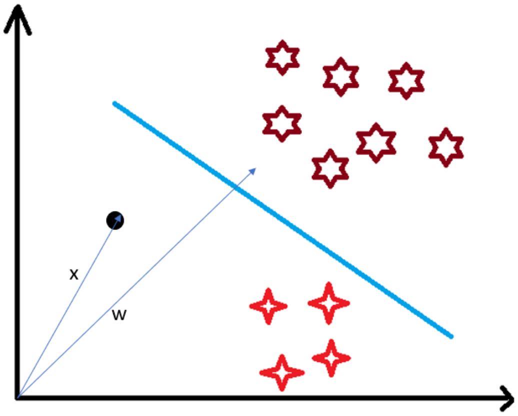


The best hyperplane is that plane that has the maximum distance from both the classes, and this is the main aim of SVM. This is done by finding different hyperplanes which classify the labels in the best way then it will choose the one which is farthest from the data points or the one which has a maximum margin.

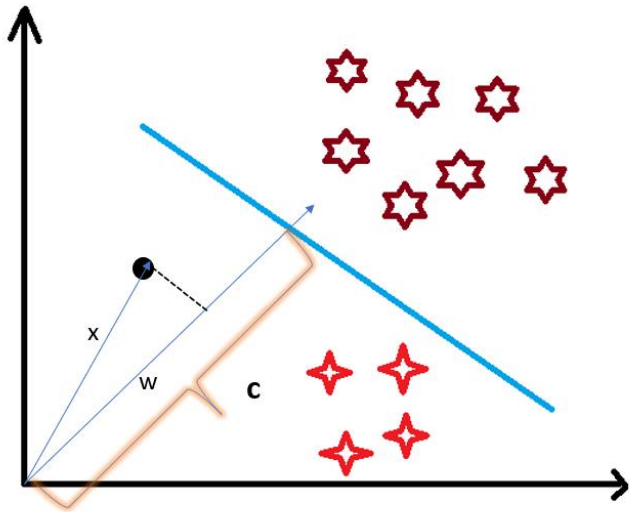


Use of Dot Product in SVM

Consider a random point X and we want to know whether it lies on the right side of the plane or the left side of the plane (positive or negative).



To find this first we assume this point is a vector (X) and then we make a vector (w) which is perpendicular to the hyperplane. Let's say the distance of vector w from origin to decision boundary is 'c'. Now we take the projection of X vector on w.



We already know that projection of any vector or another vector is called dot-product. Hence, we take the dot product of x and w vectors. If the dot product is greater than 'c' then we can say that the point lies on the right side. If the dot product is less than 'c' then the point is on the left side and if the dot product is equal to 'c' then the point lies on the decision boundary.

$$\overrightarrow{X}.\overrightarrow{w} \;= c \;\; (the\ point\ lies\ on\ the\ decision\ boundary)$$

$$\overrightarrow{X}.\overrightarrow{w} \; > \; c \; (positive\ samples)$$

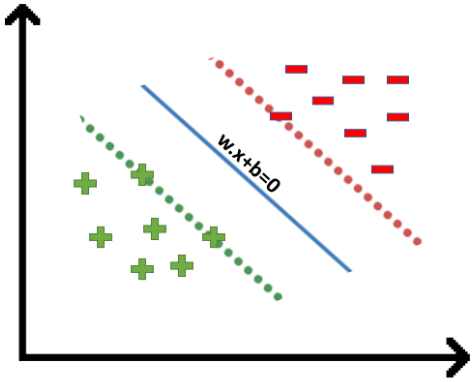
$$\overrightarrow{X}.\overrightarrow{w} \; < \; c \; (negative\ samples)$$

You must be having this doubt that why did we take this perpendicular vector w to the hyperplane? So what we want is the distance of vector X from the decision boundary and there can be infinite points on the boundary to measure the distance from. So that's why we come to standard, we simply take perpendicular and use it as a reference and then take projections of all the other data points on this perpendicular vector and then compare the distance.

In SVM we also have a concept of margin. In the next section, we will see how we find the equation of a hyperplane and what exactly do we need to optimize in SVM.

Margin in Support Vector Machine

We all know the equation of a hyperplane is w.x+b=0 where w is a vector normal to hyperplane and b is an offset.



To classify a point as negative or positive we need to define a decision rule. We can define decision rule as:

$$\overrightarrow{X}.\overrightarrow{w} - \mathrm{c} \geq 0$$

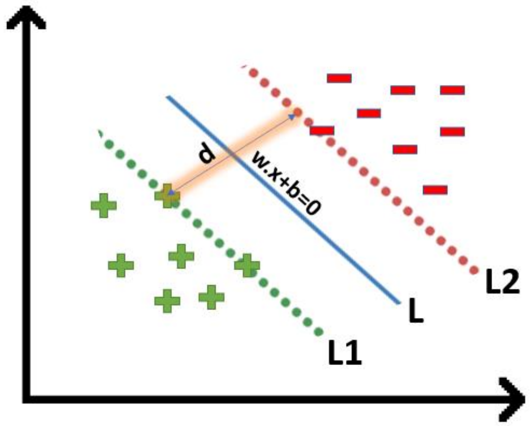
putting $-\mathrm{c}$ as b, we get

$$\overrightarrow{X}.\overrightarrow{w} + \mathrm{b} \geq 0$$

hence

$$\mathrm{y} = \begin{cases} +1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + \mathrm{b} \geq 0 \\ -1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + \mathrm{b} < 0 \end{cases}$$

If the value of w.x+b>0 then we can say it is a positive point otherwise it is a negative point. Now we need (w,b) such that the margin has a maximum distance. Let’s say this distance is ‘d’.



To calculate ‘d’ we need the equation of L1 and L2. For this, we will take few assumptions that the equation of L1 is **w.x+b=1** and for L2 it is **w.x+b=-1**.

Optimization Function and its Constraints

In order to get our optimization function, there are few constraints to consider. That constraint is that **“We’ll calculate the distance (d) in such a way that no positive or negative point can cross the margin line”.** Let’s write these constraints mathematically:

$$For\ all\ the\ Red\ points\ \overrightarrow{w}.\overrightarrow{X}+b \leq -1$$

$$For\ all\ the\ Green\ points\ \overrightarrow{w}.\overrightarrow{X}+b \geq 1$$

Rather than taking 2 constraints forward, we’ll now try to simplify these two constraints into 1. We assume that negative classes have ***y=-1*** and positive classes have ***y=1.***

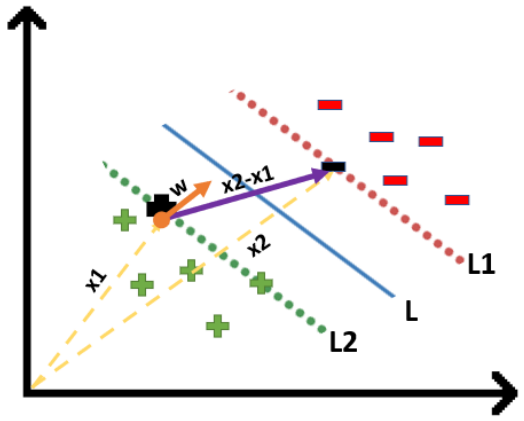
We can say that for every point to be correctly classified this condition should always be true:

$$y_i(\overrightarrow{w}.\overrightarrow{X}+b) \geq 1$$

Suppose a green point is correctly classified that means it will follow **w.x+b>=1,** if we multiply this with **y=1** we get this same equation mentioned above. Similarly, if we do this with a red point with **y=-1** we will again get this equation**.** Hence, we can say that we need to maximize (d) such that this constraint holds true.

We will take 2 support vectors, 1 from the negative class and 2nd from the positive class. The distance between these two vectors x1 and x2 will be *(x2-x1) vector*. What we need is, the shortest distance between these two points which can be found using a trick we used in the dot product.

We take a vector ‘w’ perpendicular to the hyperplane and then find the projection of (x2-x1) vector on ‘w’. **Note:** this perpendicular vector should be a unit vector then only this will work. Why this should be a unit vector? This has been explained in the dot-product section. To make this ‘w’ a unit vector we divide this with the norm of ‘w’.



Finding Projection of a Vector on Another Vector Using Dot Product

We already know how to find the projection of a vector on another vector. We do this by dot-product of both vectors. So let’s see how

$$\Rightarrow \quad (\mathrm{x2}-\mathrm{x1}) \,.\, \frac{\overrightarrow{\mathrm{w}}}{\|\mathrm{w}\|}$$

$$\Rightarrow \quad \frac{\mathrm{x2}\,.\,\overrightarrow{\mathrm{w}} - \ \mathrm{x1}\,.\,\overrightarrow{\mathrm{w}}}{\|\mathrm{w}\|} \qquad ----(1)$$

Since x2 and x1 are support vectors and they lie on the hyperplane, hence they will follow **y_i* (2.x+b)=1** so we can write it as:

$$\text{for positive point } \mathrm{y} = 1$$

$$\Rightarrow \; 1 \times (\overrightarrow{\mathrm{w}} . x1 + b) \; = \; 1$$

$$\Rightarrow \; \overrightarrow{\mathrm{w}} . x1 \; = \; 1 - \mathrm{b} \qquad - - - - - (2)$$

$$\text{Similarly for negative point } \mathrm{y} = -1$$

$$\Rightarrow \; -1 \times \left(\overrightarrow{\mathrm{w}} . x2 + b \right) = \; 1$$

$$\Rightarrow \overrightarrow{\mathrm{w}} . x2 \; = -\mathrm{b} - 1 \qquad - - - - - (3)$$

Putting equations (2) and (3) in equation (1) we get:

$$\Rightarrow \; \frac{(1 - \mathrm{b}) - (-\mathrm{b} - 1)}{\|\mathrm{w}\|}$$

$$\Rightarrow \; \frac{1 - \mathrm{b} + \mathrm{b} + 1}{\|\mathrm{w}\|} \; = \; \frac{2}{\|\mathrm{w}\|} \; = \mathrm{d}$$

Hence the equation which we have to maximize is:

$$\mathrm{argmax}(\mathrm{w}^{*}, \mathrm{b}^{*}) \; \frac{2}{\|\mathrm{w}\|} \text{ such that } \mathrm{y}_{\mathrm{i}} \left(\overrightarrow{w} . \overrightarrow{X} + b \right) \geq 1$$

We have now found our optimization function but there is a catch here that we don’t find this type of perfectly linearly separable data in the industry, there is hardly any case we get this type of data and hence we fail to use this condition we proved here. The type of problem which we just studied is called **Hard Margin SVM** now we shall study soft margin which is similar to this but there are few more interesting tricks we use in **Soft Margin SVM.**

Soft Margin SVM

In real-life applications, we rarely encounter datasets that are perfectly linearly separable. Instead, we often come across datasets that are either nearly linearly separable or entirely non-linearly separable. Unfortunately, the trick demonstrated above for linearly separable datasets is not applicable in these cases. This is where Support Vector Machines (SVM) come into play. These are a powerful tool in machine learning that can effectively handle both almost linearly separable and non-linearly separable datasets, providing a robust solution to classification problems in diverse real-world scenarios.

To tackle this problem what we do is modify that equation in such a way that it allows few misclassifications that means it allows few points to be wrongly classified.

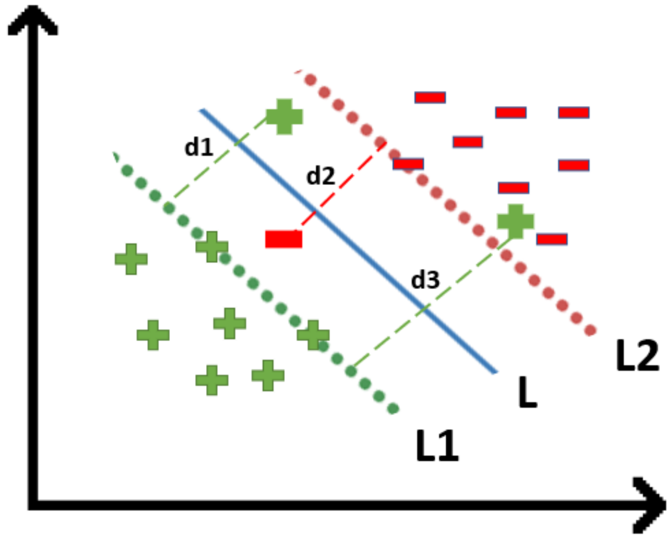
We know that *max[f(x)]* can also be written as *min[1/f(x)]*, it is common practice to minimize a cost function for optimization problems; therefore, we can invert the function.

$$\mathrm{argmin}(\mathrm{w}^*, \mathrm{b}^*)\ \frac{\|\mathrm{w}\|}{2}\ \text{such that}\ \mathrm{y}_\mathrm{i}\left(\vec{w}.\vec{X}+b\right) \geq 1$$

To make a soft margin equation we add 2 more terms to this equation which is **zeta** and multiply that by a **hyperparameter 'c'**

$$\mathrm{argmin}\left(\mathrm{w}^*, \mathrm{b}^*\right)\ \frac{\|\mathrm{w}\|}{2}\ + c\sum_{i=1}^{n}\zeta_i$$

For all the ***correctly classified*** points our **zeta** will be equal to 0 and for all the ***incorrectly classified*** points the **zeta** is simply the distance of that particular point from its correct hyperplane that means if we see the wrongly classified green points the value of **zeta** will be the distance of these points from L1 hyperplane and for wrongly classified redpoint **zeta** will be the distance of that point from L2 hyperplane.



So now we can say that our that are **SVM Error = Margin Error + Classification Error.** The higher the margin, the lower would-be margin error, and vice versa.

Let's say you take a high value of 'c' =1000, this would mean that you don't want to focus on margin error and just want a model which doesn't misclassify any data point.

Look at the figure below:

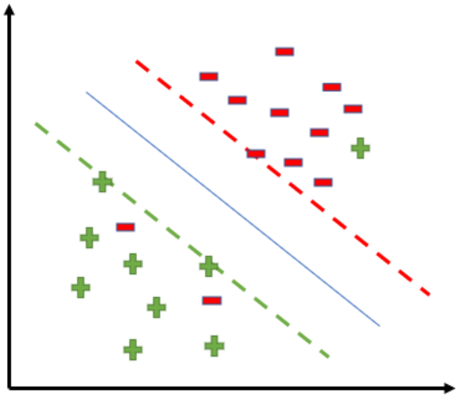


Figure 1

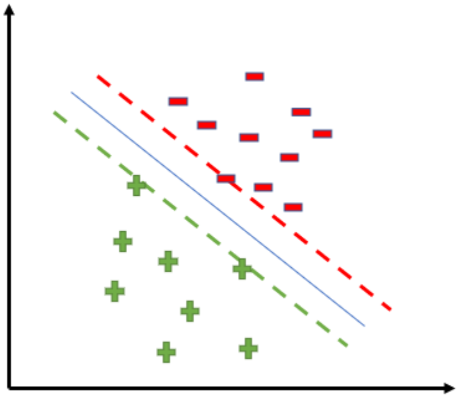


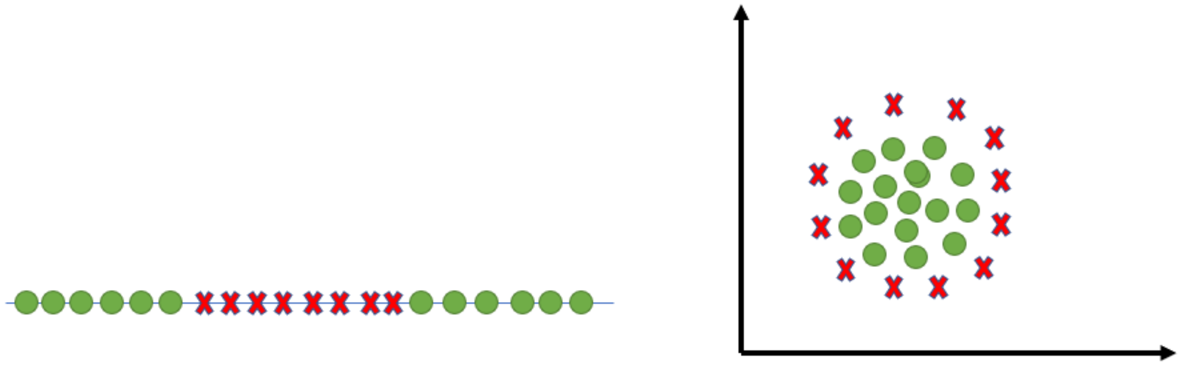
Figure 2

If someone asks you which is a better model, the one where the margin is maximum and has 2 misclassified points or the one where the margin is very less, and all the points are correctly classified?

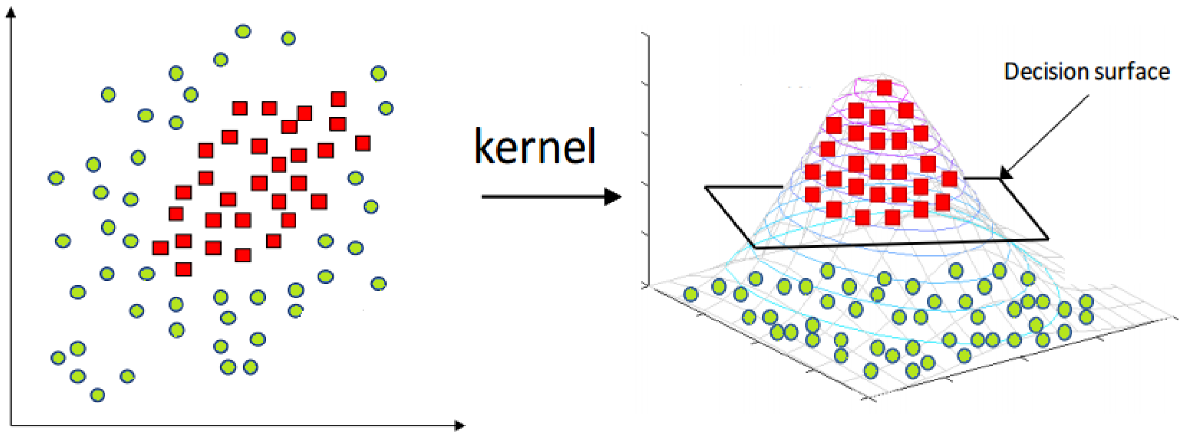
Well, there's no correct answer to this question, but rather we can use *SVM Error = Margin Error + Classification Error to* justify this. If you don't want any misclassification in the model then you can choose *figure 2*. That means we'll increase 'c' to decrease Classification Error but if you want that your margin should be maximized then the value of 'c' should be minimized. That's why ‘c’ is a hyperparameter and we find the optimal value of ‘c’ using GridsearchCV and cross-validation.

Kernels in Support Vector Machine

The most interesting feature of SVM is that it can even work with a non-linear dataset and for this, we use “Kernel Trick” which makes it easier to classifies the points. Suppose we have a dataset like this:



Here we see we cannot draw a single line or say hyperplane which can classify the points correctly. So what we do is try converting this lower dimension space to a higher dimension space using some quadratic functions which will allow us to find a decision boundary that clearly divides the data points. These functions which help us do this are called Kernels and which kernel to use is purely determined by hyperparameter tuning.



Different Kernel Functions

Some kernel functions which you can use in SVM are given below:

1. Polynomial Kernel

Following is the formula for the polynomial kernel:

$$f(X1, X2) \;=\; (X1^T . X2 + 1)^d$$

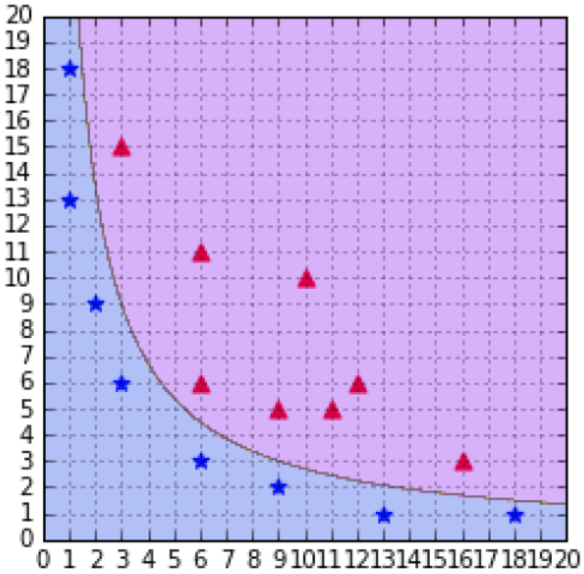
Here d is the degree of the polynomial, which we need to specify manually.

Suppose we have two features X1 and X2 and output variable as Y, so using polynomial kernel we can write it as:

$$X1^T . X2 = \begin{bmatrix} X1 \\ X2 \end{bmatrix} . \begin{bmatrix} X1 & X2 \end{bmatrix}$$

$$= \begin{bmatrix} X1^2 & X1 . X2 \\ X1 . X2 & X2^2 \end{bmatrix}$$

So we basically need to find X_1^2 , X_2^2 and X1.X2, and now we can see that 2 dimensions got converted into 5 dimensions.



A SVM using a polynomial kernel is able to separate the data (degree=2)

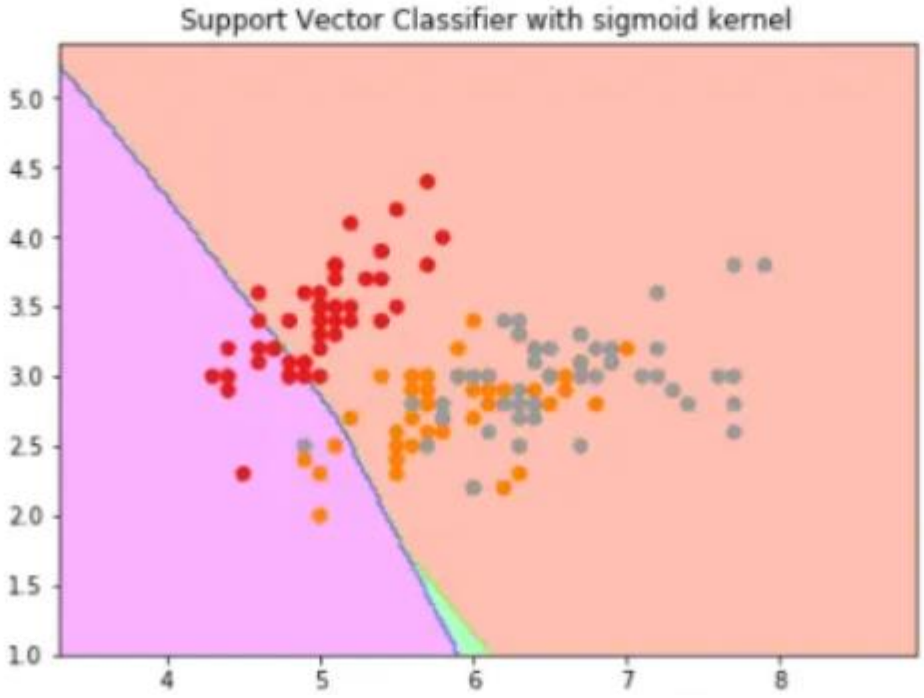
Image 4

2. Sigmoid Kernel

We can use it as the proxy for neural networks. Equation is:

$$f(x1, x2) \ = \ \mathrm{tan}h(\alpha x^T y + x)$$

It is just taking your input, mapping them to a value of 0 and 1 so that they can be separated by a simple straight line.



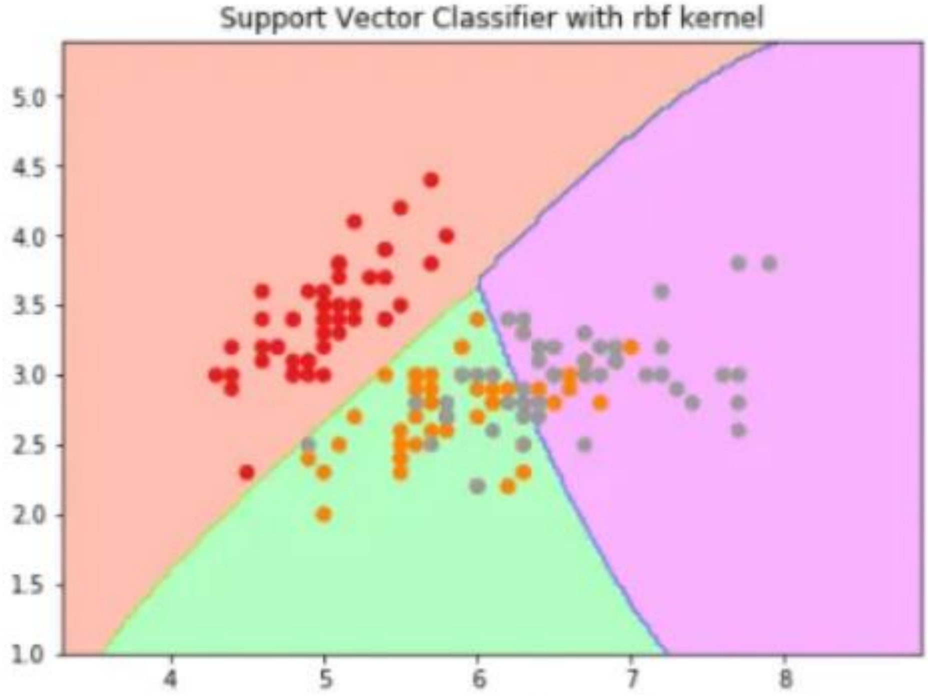
3. RBF Kernel

What it actually does is to create non-linear combinations of our features to lift your samples onto a higher-dimensional feature space where we can use a linear decision boundary to separate your classes It is the most used kernel in SVM classifications, the following formula explains it mathematically:

$$f(x1, x2) = e^{\frac{-||(x1 - x2)||^2}{2\sigma^2}}$$

where,

1. 'σ' is the variance and our hyperparameter
2. $||X_1 - X_2||$ is the Euclidean Distance between two points X_1 and X_2



4. Bessel function kernel

It is mainly used for eliminating the cross term in mathematical functions. Following is the formula of the Bessel function kernel:

$$k(x, y) = \frac{J_{v+1}(\sigma||x - y||)}{||x - y||^{-n(v+1)}}$$

5. Anova Kernel

It performs well on multidimensional regression problems. The formula for this kernel function is:

$$k(x, y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

How to Choose the Right Kernel?

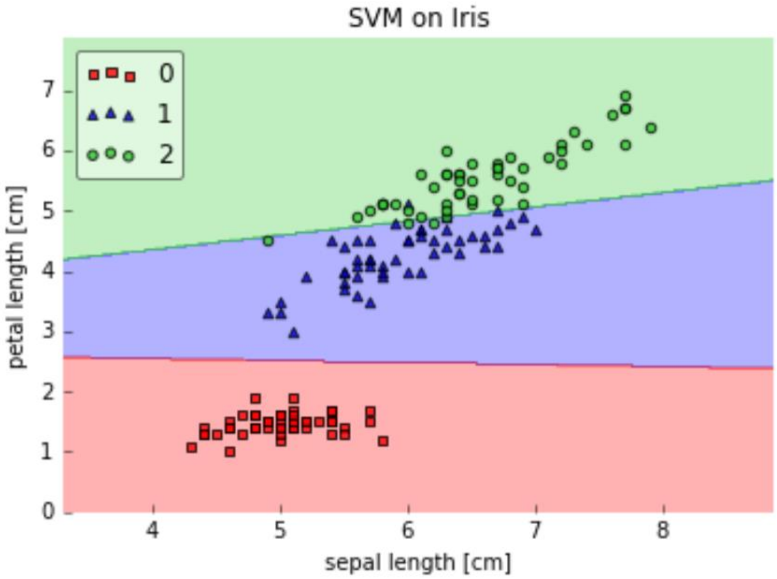
I am well aware of the fact that you must be having this doubt about how to decide which kernel function will work efficiently for your dataset. **It is necessary to choose a good kernel function because the performance of the model depends on it.**

Choosing a kernel totally depends on what kind of dataset are you working on. If it is linearly separable then you must opt. for linear kernel function since it is very easy to use and the complexity is much lower compared to other kernel functions. I'd recommend you start with a hypothesis that your data is linearly separable and choose a linear kernel function.

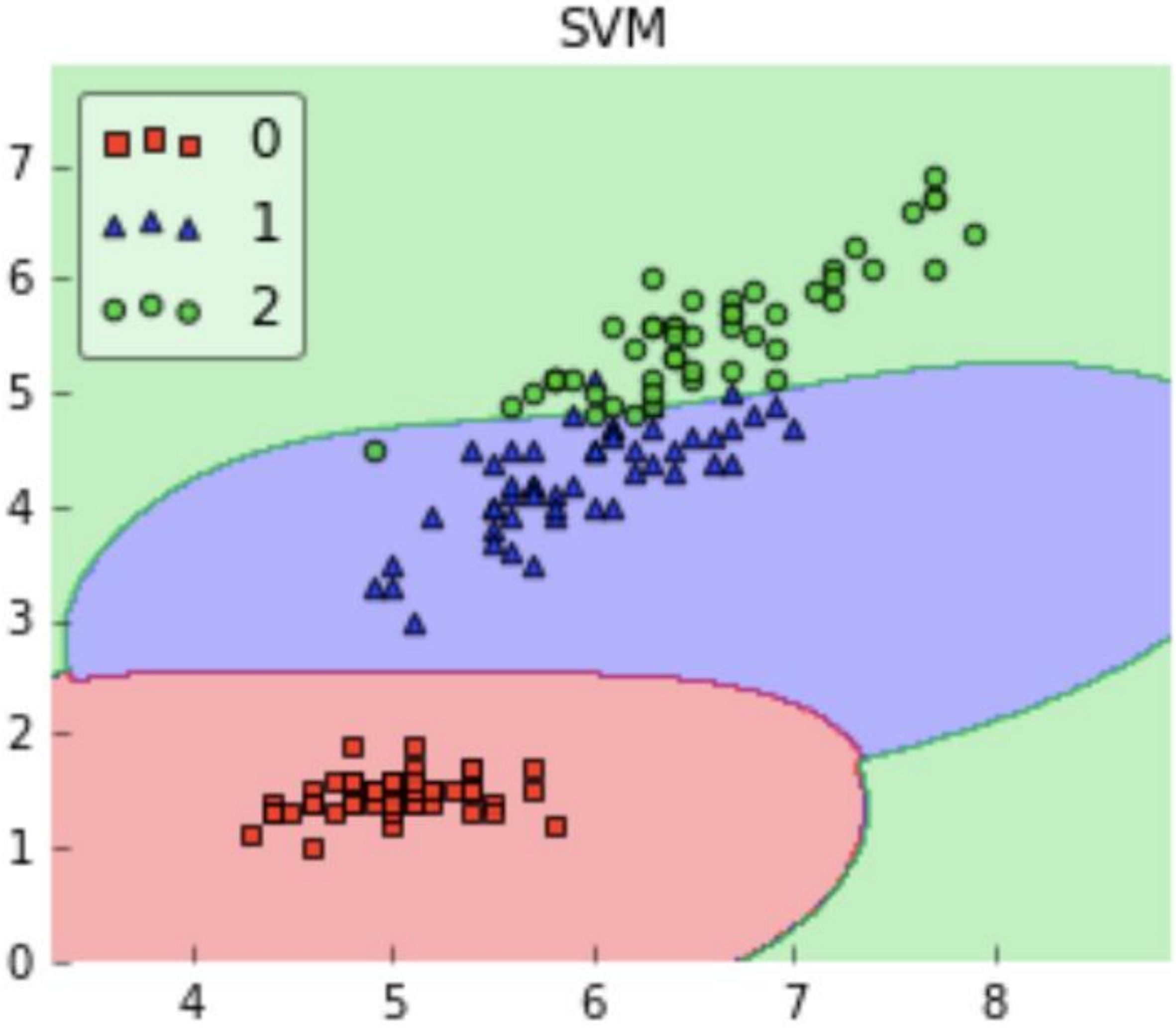
You can then work your way up towards the more complex kernel functions. Usually, we use **SVM with RBF** and linear kernel function because other kernels like polynomial kernel are rarely used due to poor efficiency. But what if linear and RBF both give approximately similar results? Which kernel do we choose now?

Example

Let's understand this with the help of an example, for simplicity I'll only take 2 features that mean 2 dimensions only. In the figure below I have plotted the decision boundary of a linear SVM on 2 features of the iris dataset:

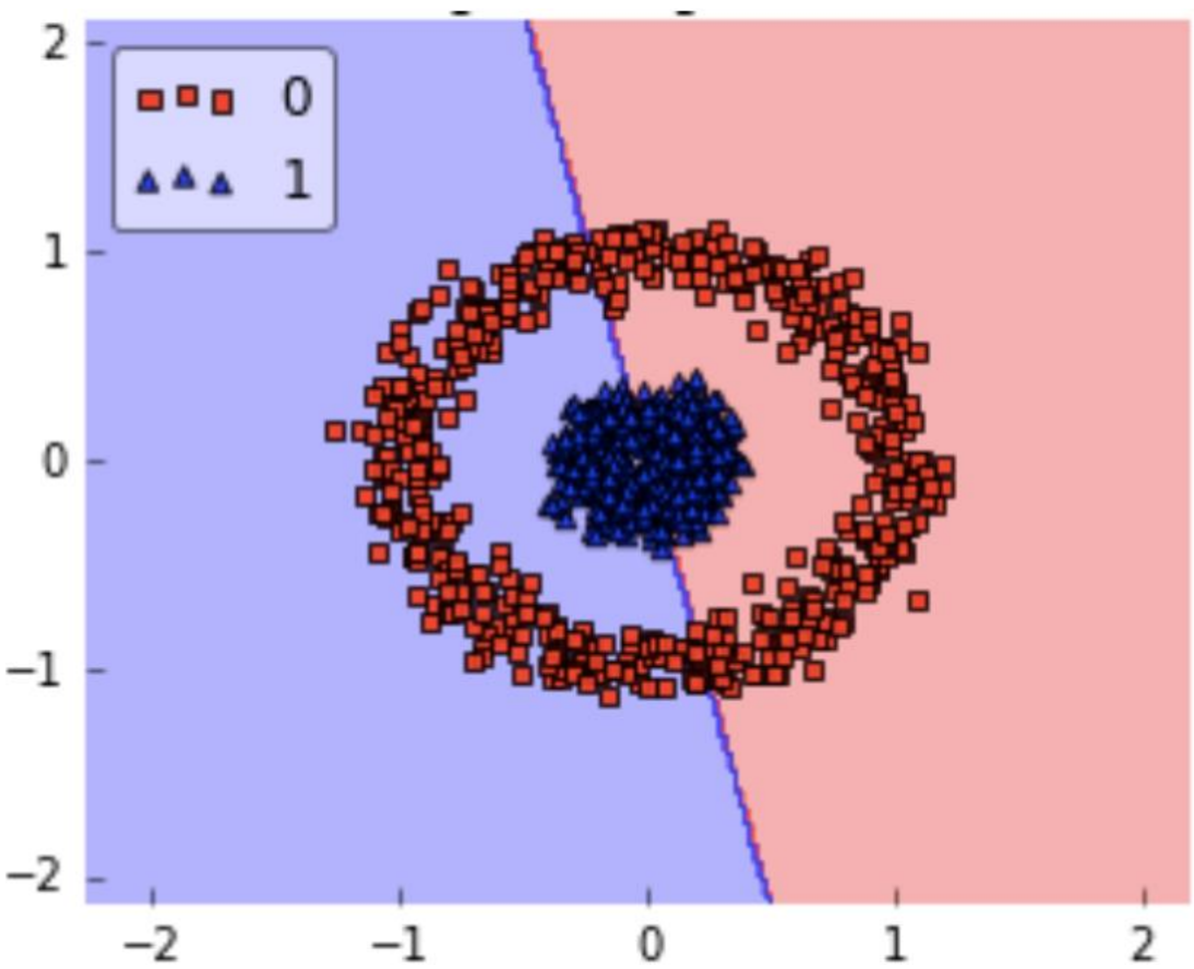


Here we see that a linear kernel works fine on this dataset, but now let’s see how will RBF kernel work.

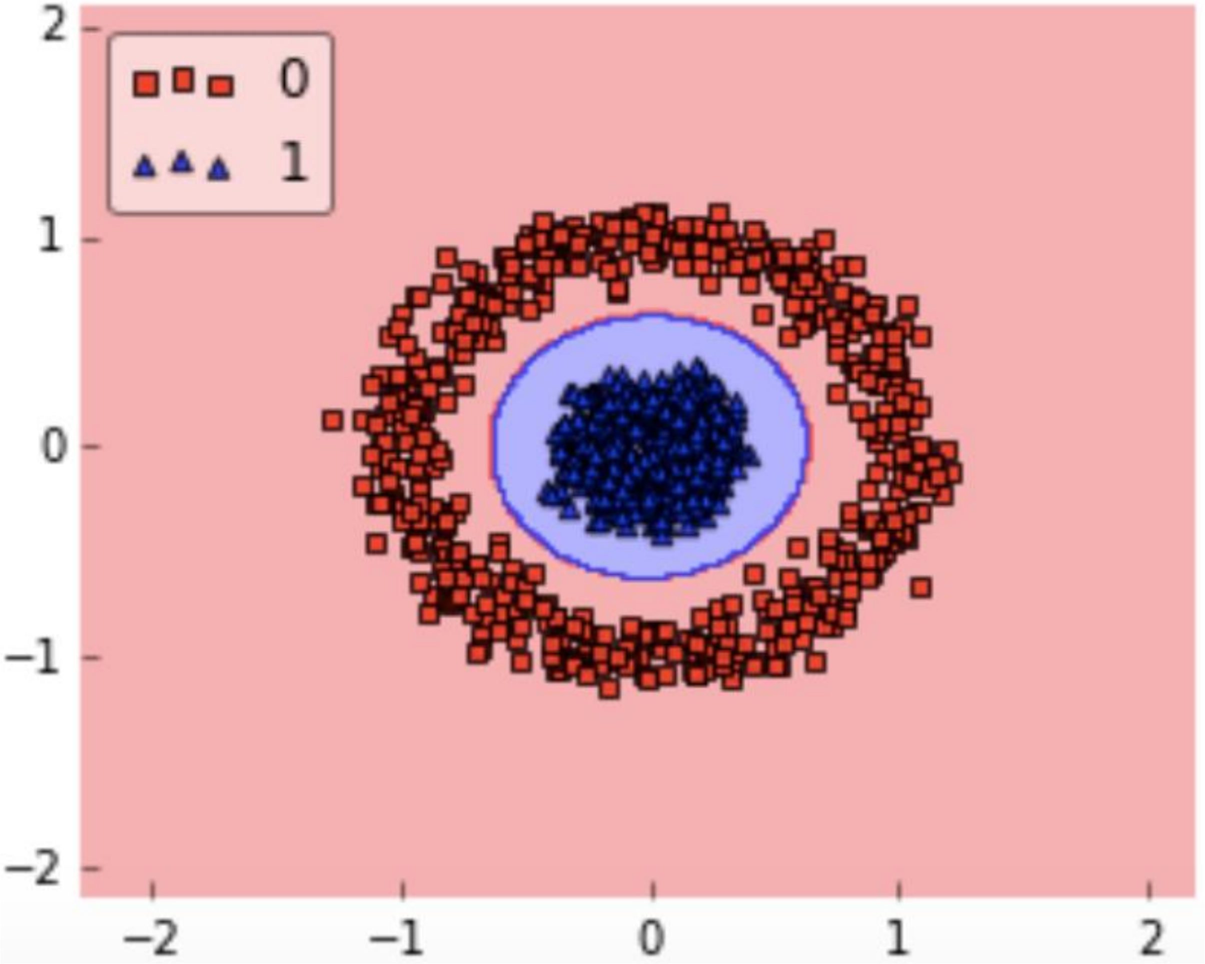


We can observe that both the kernels give similar results, both work well with our dataset but which one should we choose? Linear SVM is a parametric model. A Parametric Model is a concept used to describe a model in which all its data is represented within its parameters. In short, the only information needed to predict the future from the current value is the parameters.

The complexity of the RBF kernel grows as the training data size increases. In addition to the fact that it is more expensive to prepare RBF kernel, we also have to keep the kernel matrix around, and the projection into this “infinite” higher dimensional space where the data becomes linearly separable is more expensive as well during prediction. If the dataset is not linear then using linear kernel doesn’t make sense we’ll get a very low accuracy if we do so.



So for this kind of dataset, we can use RBF without even a second thought because it makes decision boundary like this:



Implementation and hyperparameter tuning of Support Vector Machine in Python.

Radial Basis Functions

Radial Basis Kernel is a kernel function that is used in machine learning to find a non-linear classifier or regression line. **What is Kernel Function? Kernel Function is used to transform n-dimensional input to m-dimensional input, where m is much higher than n then find the dot product in higher dimensional efficiently**. The main idea to use kernel is: A linear classifier or regression curve in higher dimensions becomes a Non-linear classifier or regression curve in lower dimensions. Mathematical Definition of Radial Basis Kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

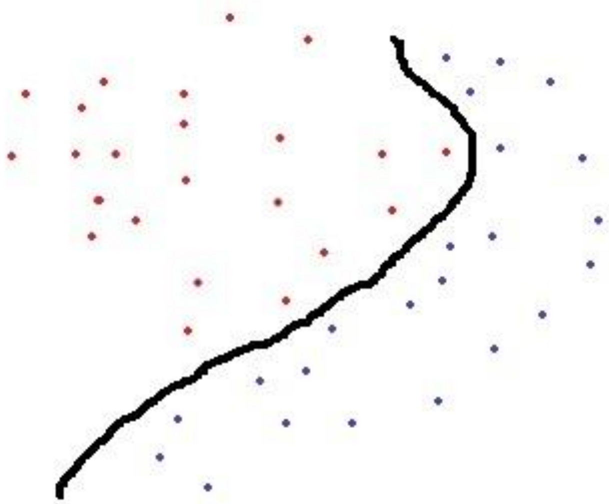
Radial Basis Kernel

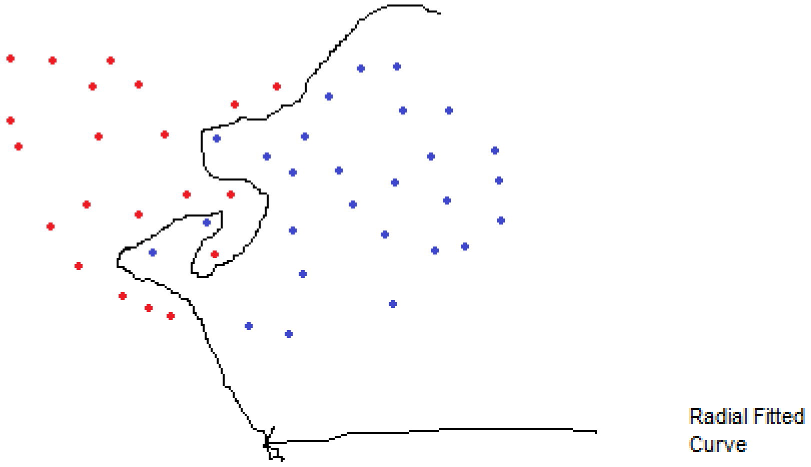
where x, x’ are vector point in any fixed dimensional space. But if we expand the above exponential expression, It will go upto infinite power of x and x’, as expansion of ex contains infinite terms upto infinite power of x hence it involves terms upto infinite powers in infinite dimension. If we apply any of the algorithms like perceptron Algorithm or linear regression on this kernel, actually we would be applying our algorithm to new infinite-dimensional datapoint we have created. Hence it will give a hyperplane in infinite dimensions, which will give a very strong non-linear classifier or regression curve after returning to our original dimensions.

$$a_1 X^{inf} + a_2 X^{inf-1} + a_3 X^{inf-2} + \dots . + a_{inf} X + C$$

polynomial of infinite power

So, Although we are applying linear classifier/regression it will give a non-linear classifier or regression line, that will be a polynomial of infinite power. And being a polynomial of infinite power, Radial Basis kernel is a very powerful kernel, which can give a curve fitting any complex dataset. Why Radial Basis Kernel Is much powerful? The main motive of the kernel is to do calculations in any d-dimensional space where d > 1, so that we can get a quadratic, cubic or any polynomial equation of large degree for our classification/regression line. Since Radial basis kernel uses exponent and as we know the expansion of e^x gives a polynomial equation of infinite power, so using this kernel, we make our regression/classification line infinitely powerful too. Some Complex Dataset Fitted Using RBF Kernel easily:





- the curse of dimensionality.

Pruning in Machine Learning

Introduction

Pruning is a technique in machine learning that involves diminishing the size of a prepared model by eliminating some of its parameters. The objective of pruning is to make a smaller, faster, and more effective model while maintaining its accuracy. Pruning can be especially useful for huge and complex models, where lessening their size can prompt significant improvements in their speed and proficiency.

Types of Pruning Techniques:

There are two principal types of pruning techniques: unstructured and structured pruning. Unstructured pruning involves eliminating individual parameters or connections from the model, resulting in a smaller and sparser model. Structured pruning involves eliminating groups of parameters, such as whole filters, channels, or neurons.

Structured Pruning:

Structured pruning involves eliminating whole structures or groups of parameters from the model, such as **whole neurons, channels, or filters.** This sort of pruning preserves the hidden structure of the model, implying that the pruned model will have the same overall architecture as the first model, but with fewer parameters.

Structured pruning is suitable for models with a structured architecture, such as convolutional neural networks (CNNs), where the parameters are coordinated into filters, channels, and layers. It is also easier to carry out than unstructured pruning since it preserves the structure of the model.

Unstructured Pruning:

Unstructured pruning involves eliminating individual parameters from the model without respect **for their location in** the model. This sort of pruning does not preserve the hidden structure of the model, implying that the pruned model will have an unexpected architecture in comparison to the first model. Unstructured pruning is suitable for models without a structured architecture, such as completely connected brain networks, where the parameters are coordinated into a single grid. It tends to be more effective than structured pruning since it allows for more fine-grained pruning; however, it can also be more difficult to execute.

Criteria for Selecting a Pruning Technique:

The decision of which pruning technique to use depends on several factors, such as the type of model, the accessibility of registration resources, and the degree of accuracy desired. For instance, structured pruning is more suitable for convolutional brain networks, while unstructured pruning is more pertinent for completely connected networks. The decision to prune should also consider the compromise between model size and accuracy. Other factors to consider include the complexity of the model, the size of the training information, and the performance metrics of the model.

Pruning in Decision Trees:

Pruning can also be applied to decision trees, which are a kind of machine learning model that learns a series of binary decisions based on the information features. Decision trees can turn out to be exceptionally huge and perplexing, prompting overfitting and decreased generalisation capacity. Pruning can be used to eliminate unnecessary branches and nodes from the decision tree, resulting in a smaller and simpler model that is less liable to overfit.

Advantages

- Decreased model size and complexity. Pruning can significantly diminish the quantity of parameters in a machine learning model, prompting a smaller and simpler model that is easier to prepare and convey.
- Faster inference. Pruning can decrease the computational cost of making predictions, prompting faster and more effective predictions.
- Further developed generalization. Pruning can forestall overfitting and further develop the generalization capacity of the model by diminishing the complexity of the model.
- Increased interpretability. Pruning can result in a simpler and more interpretable model, making it easier to understand and make sense of the model's decisions.

Disadvantages

- Possible loss of accuracy. Pruning can sometimes result in a loss of accuracy, especially in the event that such a large number of parameters are pruned or on the other hand in the event that pruning is not done cautiously.
- Increased training time. Pruning can increase the training season of the model, especially assuming it is done iteratively during training.
- Trouble in choosing the right pruning technique. Choosing the right pruning technique can be testing and may require area expertise and experimentation.
- Risk of over-pruning. Over-pruning can prompt an overly simplified model that is not accurate enough for the task.

Pruning vs Other Regularization Techniques:

1. Pruning is one of numerous regularization techniques used in machine learning to forestall overfitting and further develop the generalization capacity of the model.
2. Other famous regularization techniques incorporate L1 and L2 regularization, dropout, and early stopping.
3. Contrasted with other regularization techniques, pruning has the upside of diminishing the model size and complexity, prompting faster inference and further developed interpretability.

4. Be that as it may, pruning can also have a higher computational cost during training, and its effect on the model's performance can be less unsurprising than other regularization techniques.

Practical Considerations for Pruning

- **Choose the right pruning technique:**

The decision of pruning technique depends on the specific characteristics of the model and the task within reach. Structured pruning is suitable for models with a structured architecture, while unstructured pruning is suitable for models without a structured architecture.

- **Decide the pruning rate:**

The pruning rate determines the proportion of parameters to be pruned. It should be chosen cautiously to adjust the reduction in model size with the loss of accuracy.

- **Evaluate the effect on the model's performance:**

The effect of pruning on the model's accuracy should be evaluated using fitting metrics, such as validation accuracy or test accuracy.

- **Consider iterative pruning:**

Iterative pruning involves pruning the model on various occasions during training, which can prompt improved results than a single pruning toward the finish of training.

- **Consolidate pruning with other regularization techniques:**

Pruning can be joined with other regularization techniques, such as L1 and L2 regularization or dropout, to further develop the model's performance further.

- **Beware of over-pruning:**

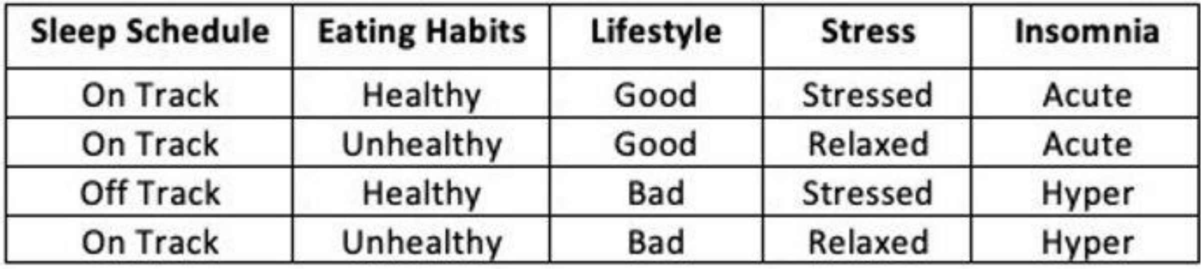
Over-pruning can prompt an overly simplified model that is not accurate enough for the task. Cautious attention should be given to choosing the right pruning rate and assessing the effect on the model's accuracy.

Decision Trees are used to find an answer to a complex problem.
The following methods could be used to achieve these answers:
1. Information Gain
2. Gini Index
3. Gain Ratio

Let us further understand how to calculate Information Gain and Gini Index with the help of an example.

Insomnia is a sleep disorder marked by problems getting to sleep, staying asleep and sleeping as long as one would like. It can have serious effects, leading to excessive lethargy, a higher risk of accidents and health effects from sleep deprivation.

A few major causes of Insomnia are irregular sleep schedule, unhealthy eating habits, illness and pain, bad lifestyle and stress. The following is the data we will use:

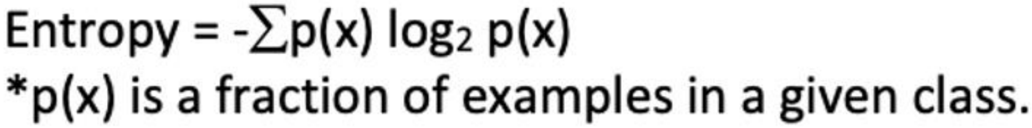


Sleep Schedule	Eating Habits	Lifestyle	Stress	Insomnia
On Track	Healthy	Good	Stressed	Acute
On Track	Unhealthy	Good	Relaxed	Acute
Off Track	Healthy	Bad	Stressed	Hyper
On Track	Unhealthy	Bad	Relaxed	Hyper

Information Gain - It is the main key that is used by decision tree Algorithms to construct it. It measures how much information a feature gives us about the class.

Information Gain = entropy(parent) – [weighted average] * entropy(children)

Entropy – It is the impurity in a group of examples. Information gain is the decrease in entropy.



$$\text{Entropy} = -\sum p(x) \log_2 p(x)$$

*p(x) is a fraction of examples in a given class.

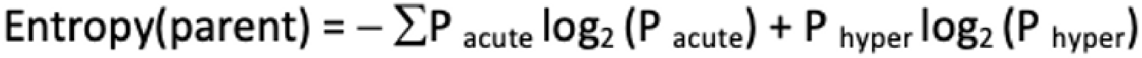
1. Analysis of Sleep Schedule feature:

Let us consider the values of the Insomnia column as the parent node (Acute, Acute, Hyper, Hyper) and the values of Sleep Schedule column as the children node (On Track, On Track, On Track, Off Track).

P(Acute) = fraction of Acute examples in parent node = 2/4 = 0.5

P(Hyper) = fraction of Hyper examples in parent node = 2/4 = 0.5

Therefore, Entropy of Parent Node:



$$\text{Entropy(parent)} = -\sum P_{\text{acute}} \log_2 (P_{\text{acute}}) + P_{\text{hyper}} \log_2 (P_{\text{hyper}})$$

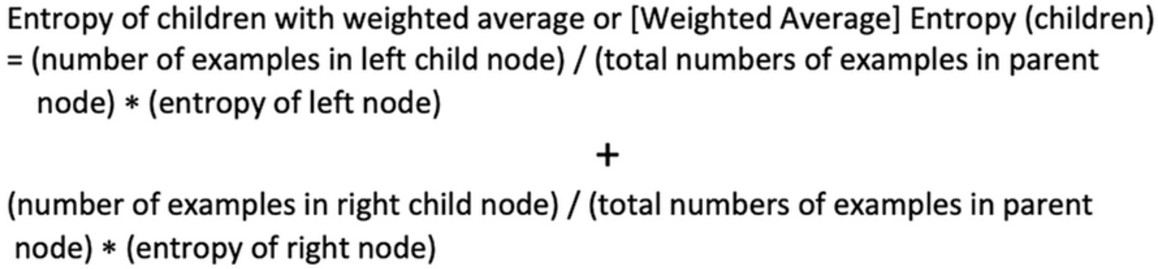
= - [0.5 log2 (0.5) + 0.5 log2 (0.5)]

= - [- 0.5 + (- 0.5)]

= 1

Entropy of left child node (On Track) = Acute, Acute, Hyper = 0.9

Entropy of right child node (Off Track) = Hyper = 0



Entropy of children with weighted average or [Weighted Average] Entropy (children)
= (number of examples in left child node) / (total numbers of examples in parent node) ∗ (entropy of left node)

+

(number of examples in right child node) / (total numbers of examples in parent node) ∗ (entropy of right node)

[Weighted Average] Entropy (children) = 3/4 * 0.9 + 1/4 * 0

= 0.675

Information Gain of Sleep Schedule

= entropy(parent) – [weighted average] * entropy(children)

= 1 – 0.675

Information Gain = 0.325

2. Analysis of Eating Habits feature:

Entropy of left child node (Healthy) is Acute, Hyper

Entropy of right child node (Unhealthy) is Acute, Hyper

Information Gain = Entropy(parent) – [Weighted average] * Entropy (children)

= 1 - (2/4 * 1 + 2/4 * 1)

= 1 - 1

Information Gain = 0

3. Analysis of Lifestyle feature:

Entropy of left child node of (Good) = Acute, Acute

Entropy of right child node of (Bad) = Hyper, Hyper

Information Gain = Entropy(parent) – [Weighted average] * Entropy (children)

= 1 - (2/4 * 0 + 2/4 * O)

= 1 - 0

Information Gain = 1

4. Analysis of Stress feature:

Entropy of left child node of (Stressed) = Acute, Hyper

Entropy of right child node of (Relaxed) = Acute, Hyper

Information Gain = Entropy(parent) – [Weighted average] * Entropy (children)

= 1 - (2/4 * 1 + 2/4 * 1)

= 1 - 1

Information Gain = 0

As per the calculations above, the information gain of Sleep Schedule is 0.325, Eating Habits is 0, Lifestyle is 1 and Stress is 0. So, the Decision Tree Algorithm will construct a decision tree based on feature that has the highest information gain. In our case it is Lifestyle, wherein the information gain is 1.

Gini Index - Gini Index or Gini Impurity is the measurement of probability of a variable being classified wrongly when it is randomly chosen. The degree of Gini Index varies from zero to one.

Formula –

$$\text{Gini Index} = 1 - \sum (P_x)^2$$

Here, (Pi) is the probability of an element classified wrongly.

1. Analysis of Sleep Schedule feature:

P (On Track) = 3/4

P (Off Track) = 1/4

Sleep Schedule = On Track and Insomnia = Acute = 2/3

Sleep Schedule = On Track and Insomnia = Hyper = 1/3

Gini Index (On Track) = 1 - [(2/3)2 + (1/3)2]
= 1 - [0.44 + 0.11]
= 0.45
Sleep Schedule = Off Track and Insomnia = Acute = 0
Sleep Schedule = Off Track and Insomnia = Hyper = 1
Gini Index (Off Track) = 1 - [02 + 12]
= 0
Weighted sum of Gini indices (Sleep Schedule)
= [3/4 * 0.45] + [1/4 * 0]
= 0.3375
2. Analysis of Eating Habits feature:
P (Healthy) = 2/4
P (Unhealthy) = 2/4
Eating Habits = Healthy and Insomnia = Acute = 1/2
Eating Habits = Healthy and Insomnia = Hyper = 1/2
Gini Index (Healthy) = 1 - [(1/2)2 + (1/2)2]
= 0.5
If (Eating Habits = Unhealthy and Insomnia = Acute): 1/2
If (Eating Habits = Unhealthy and Insomnia = Hyper): 1/2
Gini Index (Unhealthy) = 1 - [(1/2)2 + (1/2)2]
= 0.5
Weighted sum of Gini indices (Eating Habits)
= [2/4 *0.5] + [2/4*0.5]
= 0.25 + 0.25
= 0.5
3. Analysis of Lifestyle feature:
P(Good) = 2/4
P(Bad) = 2/4
Lifestyle = Good and Insomnia = Acute = 1
Lifestyle = Good and Insomnia = Hyper = 0
Gini Index (Good) = 1 - [12 + 0]
= 0
Lifestyle = Bad and Insomnia = Hyper = 0
Lifestyle = Bad and Insomnia = Hyper = 1
Gini Index(Bad) = 1 - [02 + 12]
= 0
Weighted sum of Gini indices (Lifestyle)
= [2/4 *0] + [2/4 * 0]
= 0
4. Analysis of Stress feature:
P (Stressed) = 2/4
P (Relaxed) = 2/4
Stress = Stressed and Insomnia = Acute = ½
Stress = Stressed and Insomnia = Hyper = ½
Gini Index (Stressed) = 1 - [(1/2)2 + (1/2)2] = 0.5
Stress = Relaxed and Insomnia = Acute = 1/2

Stress = Relaxed and Insomnia = Hyper = 1/2
Gini Index (Relaxed) = 1 - [(1/2)2 + (1/2)2] = 0.5
Weighted sum of Gini indices (Stress)
= [2/4 *0.5] + [2/4*0.5]
= 0.25 + 0.25
= 0.5
Therefore,
Gini index for Sleep Schedule is 0.3375
Gini index for Eating Habits is 0.5
Gini index for Lifestyle is 0
Gini index for Stress is 0.5
Since we have the Gini Index of Lifestyle to be 0, it denotes that it is pure and insomnia majorly depends upon the kind of Lifestyle a patient is leading.
Unhealthy habits and routines related to lifestyle and food and drink can increase a person's risk of insomnia. Habits like working late, uncontrolled screen time, a long afternoon nap, excessive caffeine and alcohol consumption are a few lifestyle factors can cause wakefulness.
There are many other factors like illness, pain, mental health disorders, anxiety, depression, side effects of medication, neurological problems, specific sleep disorders like restless leg syndrome.
Using Decision Tree to process data available to find out the highest determining cause of insomnia is the best choice. Decision Trees are very simple to understand because of their visual representation. They can handle a large quantity of quality data which can be validated using statistical sets and are computationally inexpensive. They are a white box type of Machine Learning algorithm as compared to the Neural Network black box. It is also a distribution free method which does not depend upon probability distribution assumptions and lastly because it can handle high dimensional data with really good accuracy.

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.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

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

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

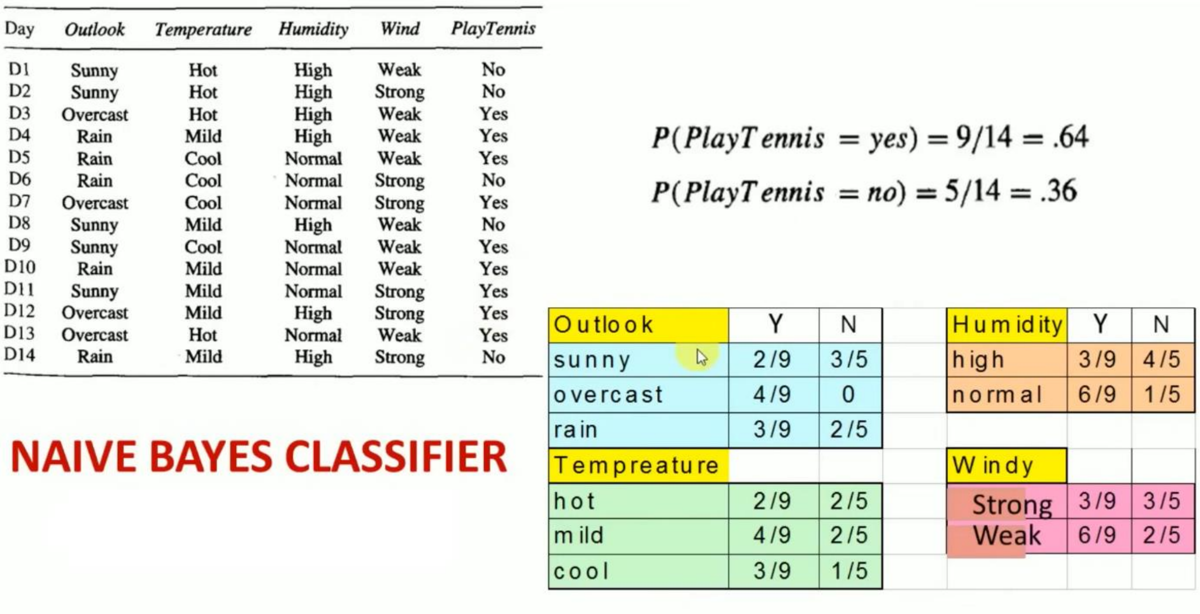
Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular

document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.

- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.



Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

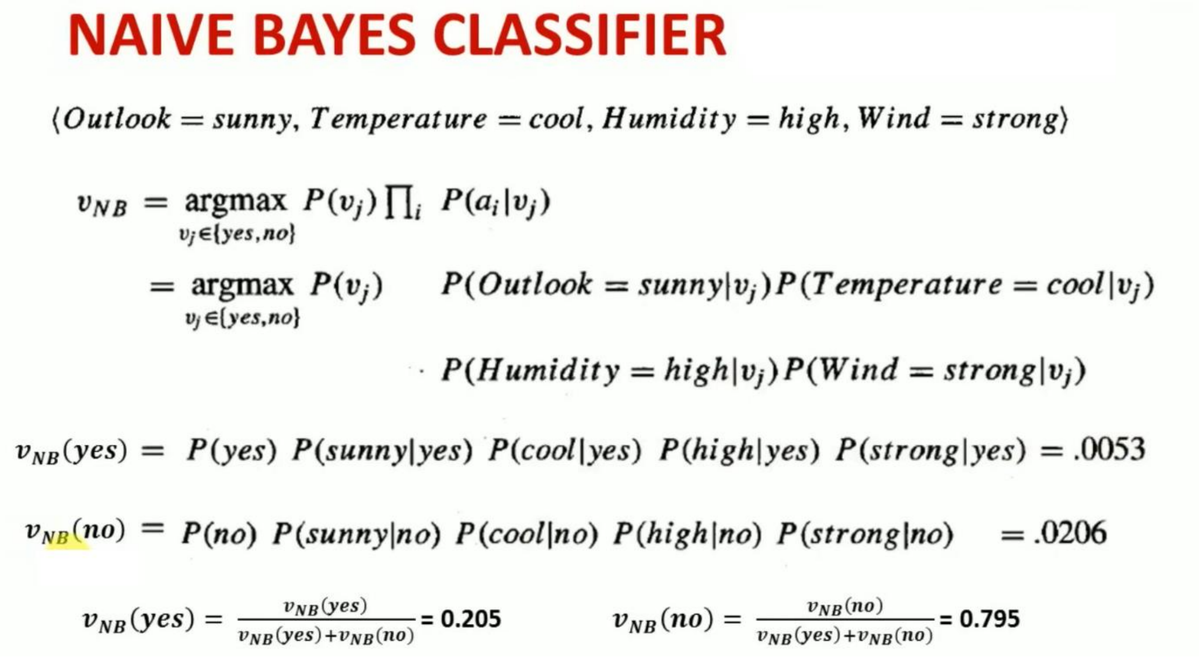
NAIVE BAYES CLASSIFIER

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5
Windy		
Strong	3/9	3/5
Weak	6/9	2/5



NAIVE BAYES CLASSIFIER

$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \prod_i P(a_i|v_j)$$

$$= \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \quad P(Outlook = sunny|v_j)P(Temperature = cool|v_j)$$

$$\cdot \; P(Humidity = high|v_j)P(Wind = strong|v_j)$$

$$v_{NB}(yes) = P(yes)\; P(sunny|yes)\; P(cool|yes)\; P(high|yes)\; P(strong|yes) = .0053$$

$$v_{NB}(no) = P(no)\; P(sunny|no)\; P(cool|no)\; P(high|no)\; P(strong|no) = .0206$$

$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes)+v_{NB}(no)} = 0.205 \qquad v_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes)+v_{NB}(no)} = 0.795$$

Radial Basis Function Kernel – Machine Learning

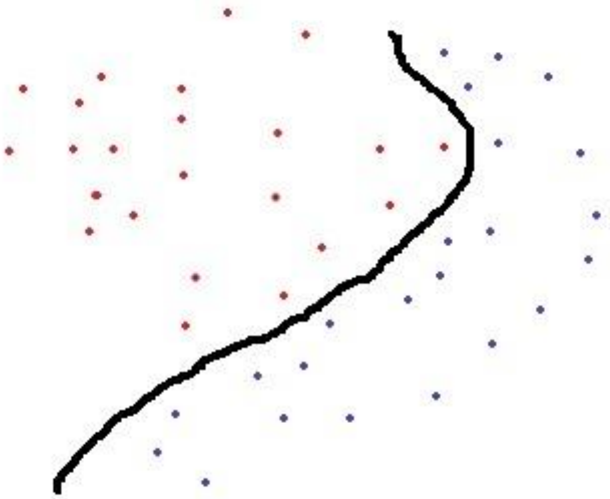
Radial Basis Kernel **is a kernel function that is used in machine learning to find a non-linear classifier or regression line.** What is Kernel Function? **Kernel Function is used to transform n-dimensional input to m-dimensional input, where m is much higher than n then find the dot product in higher dimensional efficiently. The main idea to use kernel is: A linear classifier or regression curve in higher dimensions becomes a Non-linear classifier or regression curve in lower dimensions.** Mathematical Definition of Radial Basis Kernel:

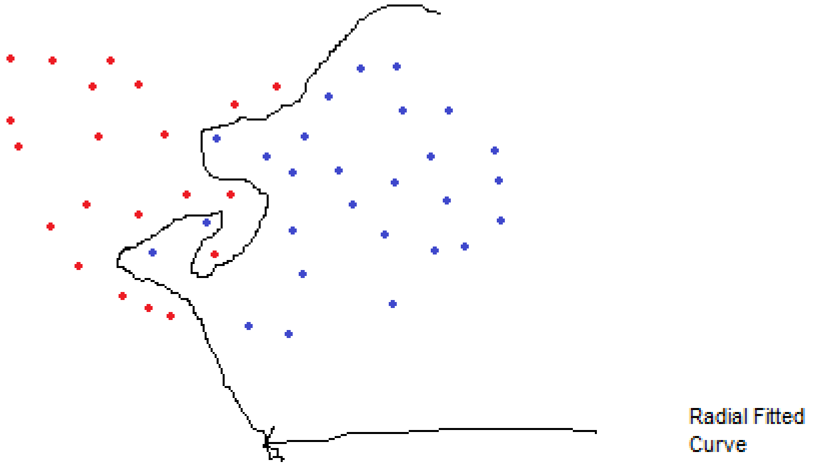
$$K(\mathbf{x}, \mathbf{x'}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x'}\|^2}{2\sigma^2}\right)$$

where *x, x'* are vector point in any fixed dimensional space. But if we expand the above exponential expression, It will go upto infinite power of *x* and *x'*, as expansion of *ex* contains infinite terms upto infinite power of *x* hence it involves terms upto infinite powers in infinite dimension. If we apply any of the algorithms like perceptron Algorithm or linear regression on this kernel, actually we would be applying our algorithm to new infinite-dimensional datapoint we have created. Hence it will give a hyperplane in infinite dimensions, which will give a very strong non-linear classifier or regression curve after returning to our original dimensions.

$$a_1 X^{inf} + a_2 X^{inf-1} + a_3 X^{inf-2} + \ldots\ldots + a_{inf} X + C$$

So, Although we are applying linear classifier/regression it will give a non-linear classifier or regression line, that will be a polynomial of infinite power. And being a polynomial of infinite power, Radial Basis kernel is a very powerful kernel, which can give a curve fitting any complex dataset. **Why Radial Basis Kernel Is much powerful?** The main motive of the kernel is to do calculations in any d-dimensional space where *d > 1*, so that we can get a quadratic, cubic or any polynomial equation of large degree for our classification/regression line. Since Radial basis kernel uses exponent and as we know the expansion of e^x gives a polynomial equation of infinite power, so using this kernel, we make our regression/classification line infinitely powerful too. **Some Complex Dataset Fitted Using RBF Kernel easily:**





Bayesian Belief Network

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.

Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

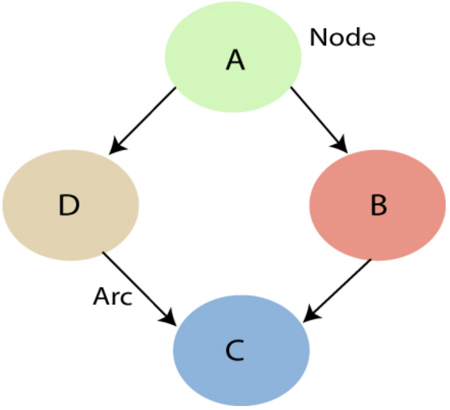
Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where

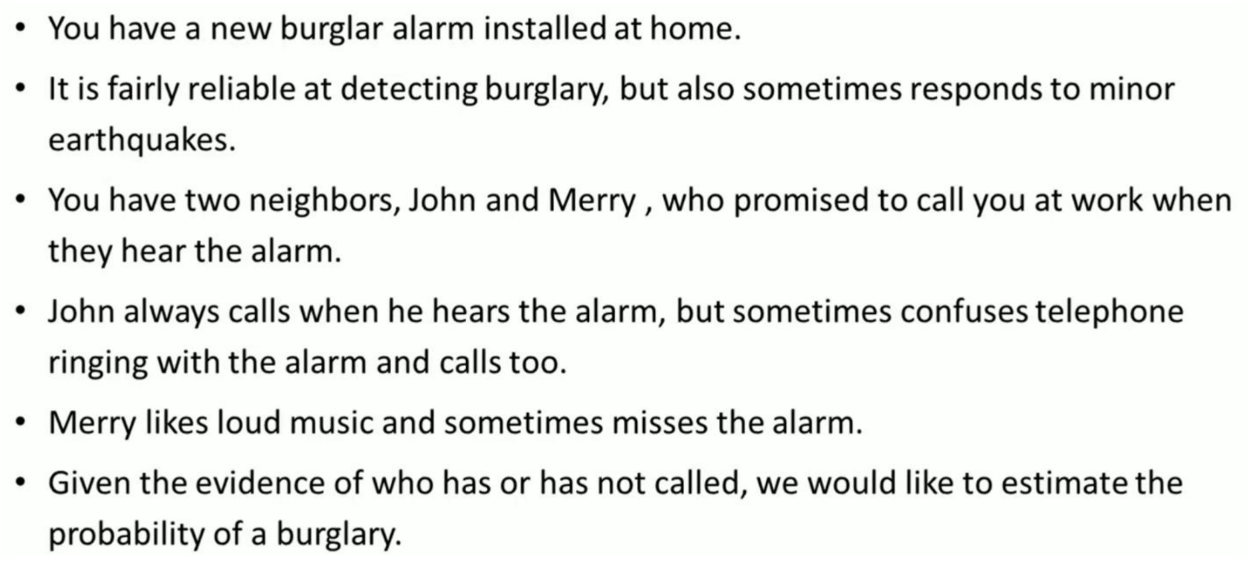


- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
  These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
  - **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
  - **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
  - **Node C is independent of node A.**

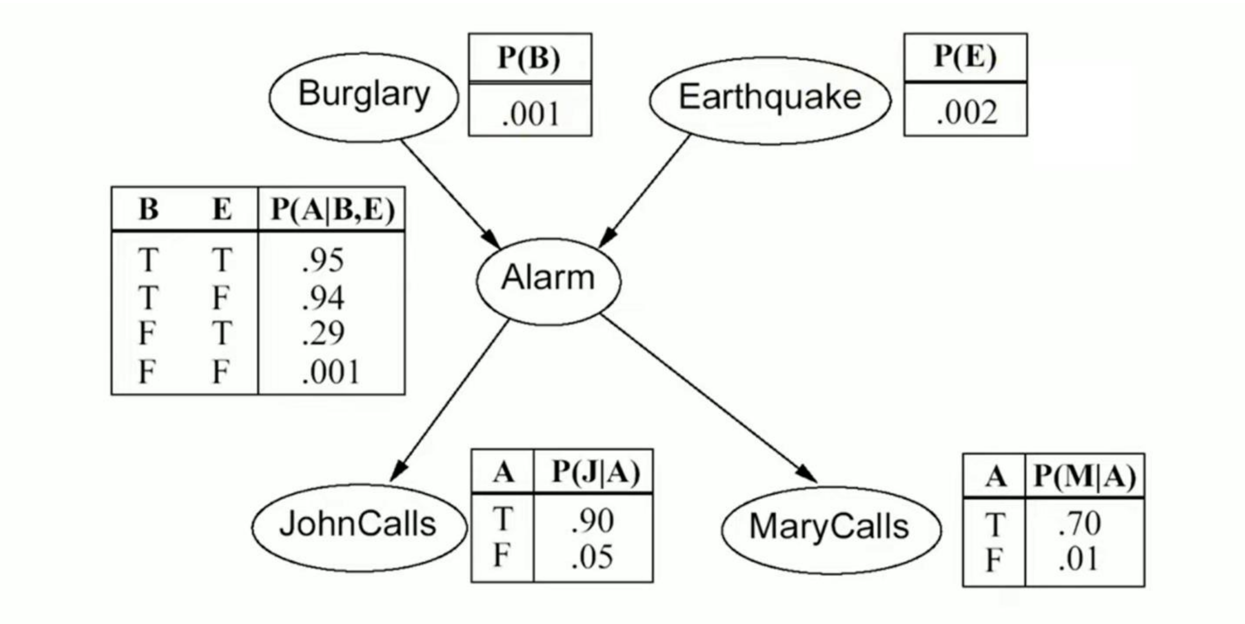
The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

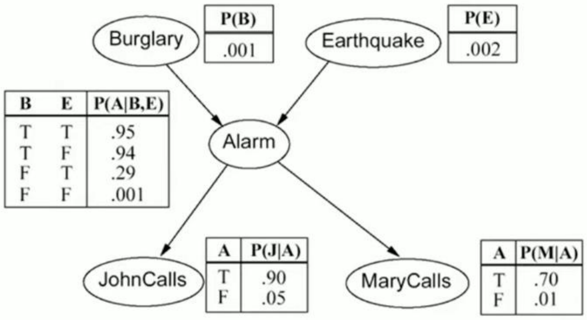
Each node in the Bayesian network has condition probability distribution **$P(X_i | Parent(X_i))$**, which determines the effect of the parent on that node.

- Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:
- Joint probability distribution:
- If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.
- **$P[x_1, x_2, x_3, \ldots, x_n]$**, it can be written as the following way in terms of the joint probability distribution.
- **$= P[x_1 | x_2, x_3, \ldots, x_n] P[x_2, x_3, \ldots, x_n]$**
- **$= P[x_1 | x_2, x_3, \ldots, x_n] P[x_2 | x_3, \ldots, x_n] \ldots P[x_{n-1} | x_n] P[x_n]$.**
- In general for each variable Xi, we can write the equation as:
- $P(X_i | X_{i-1}, \ldots, X_1) = P(X_i | Parents(X_i))$
- Explanation of Bayesian network:
- Let's understand the Bayesian network through an example by creating a directed acyclic graph:
- **Example:** Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.
- **Problem:**
- **Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.**
- 

  - You have a new burglar alarm installed at home.
  - It is fairly reliable at detecting burglary, but also sometimes responds to minor earthquakes.
  - You have two neighbors, John and Merry , who promised to call you at work when they hear the alarm.
  - John always calls when he hears the alarm, but sometimes confuses telephone ringing with the alarm and calls too.
  - Merry likes loud music and sometimes misses the alarm.
  - Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.



1. What is the probability that the alarm has sounded but neither a burglary nor an earthquake has occurred, and both John and Merry call?

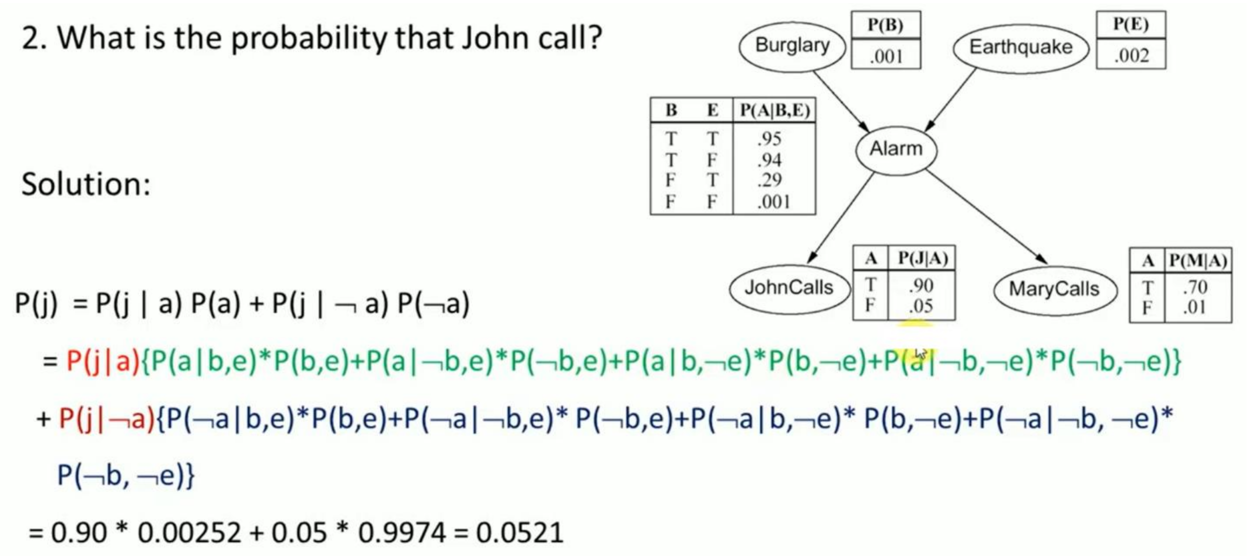


Solution:

$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) = P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$$

$$= 0.00062$$



2. What is the probability that John call?

Solution:

$P(j) = P(j \mid a)\, P(a) + P(j \mid \neg a)\, P(\neg a)$

$= P(j|a)\{P(a|b,e)*P(b,e)+P(a|\neg b,e)*P(\neg b,e)+P(a|b,\neg e)*P(b,\neg e)+P(a|\neg b,\neg e)*P(\neg b,\neg e)\}$

$+ P(j|\neg a)\{P(\neg a|b,e)*P(b,e)+P(\neg a|\neg b,e)* P(\neg b,e)+P(\neg a|b,\neg e)* P(b,\neg e)+P(\neg a|\neg b, \neg e)*$

$P(\neg b, \neg e)\}$

$= 0.90 * 0.00252 + 0.05 * 0.9974 = 0.0521$

P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01