

**SCHOOL OF COMPUTING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**UNIT IV**

**Machine Learning Tools, Techniques And Applications- SCSA3016**

**UNIT 4**

**MACHINE LEARNING TOOLS, TECHNIQUES AND APPLICATIONS**

Supervised Learning, Unsupervised Learning, Reinforcement Learning, Dimensionality Reduction, Principal Component Analysis, Classification and Regression models, Tree and Bayesian network models, Neural Networks, Testing, Evaluation and Validation of Models.

**4.1. MACHINE LEARNING**

Machine learning is **a method of data analysis that automates analytical model building**. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

Example: Image recognition, Speech recognition, Medical diagnosis, Statistical arbitrage, Predictive analytics, etc.

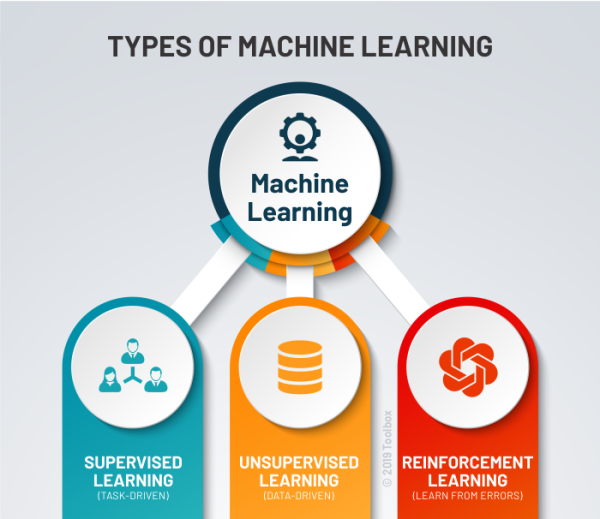
**Artificial Intelligence, Machine Learning and Deep Learning**

* AI is defined as a program that exhibits cognitive ability similar to that of a human being.
* Making computers think like humans and solve problems the way we do is one of the main tenets of artificial intelligence.
* Any computer program that shows characteristics, such as self-improvement, learning through inference, or even basic human tasks, such as image recognition and language processing, is considered to be a form of AI.
* The field of artificial intelligence includes within it the sub-fields of **machine learning and deep learning**.
* Deep Learning is a more specialized version of machine learning that utilizes more complex methods for difficult problems.
* However, the difference between machine learning and artificial intelligence is that machine learning is probabilistic (output can be explained, thereby ruling out the black box nature of AI), deep learning is deterministic.

**4.2. TYPES OF MACHINE LEARNING**

These are three types of machine learning:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

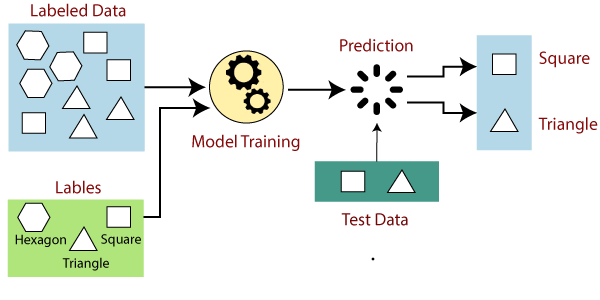


1. **Supervised Learning:**

* Supervised learning is one of the most basic types of machine learning.
* In this type, the machine learning algorithm is trained on **labelled data**.
* In supervised learning, the ML algorithm is given a small training dataset to work with.
* This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with.
* The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset.
* At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.
* This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset.
* **Example :** **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

## **How Supervised Learning Works?**

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

* If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
* If the given shape has three sides, then it will be labelled as a **triangle**.
* If the given shape has six equal sides then it will be labelled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

## **Advantages of Supervised learning:**

* With the help of supervised learning, the model can predict the output on the basis of prior experiences.
* In supervised learning, we can have an exact idea about the classes of objects.

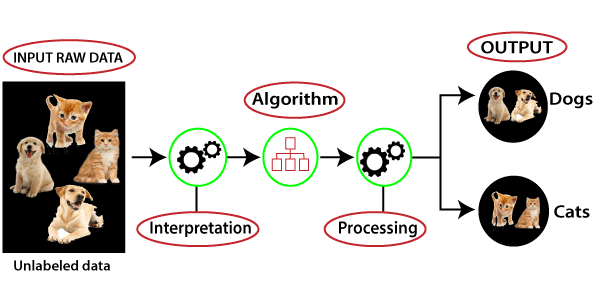
## **Disadvantages of supervised learning:**

* Supervised learning models are not suitable for handling the complex tasks.
* Supervised learning cannot predict the correct output if the test data is different from the training dataset.
* Training required lots of computation times.

1. **Unsupervised Learning:**

* Unsupervised machine learning holds the advantage of being able to work with **unlabelled data**.
* This means that human labour is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.
* In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures.
* Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.
* The creation of these hidden structures is what makes unsupervised learning algorithms versatile.
* Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures.
* This offers more post-deployment development than supervised learning algorithms.
* **Example :** Principal Component Analysis, Clustering

## **How Unsupervised Learning Works?**



Here, we have taken an unlabelled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabelled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

## **Advantages of Unsupervised Learning**

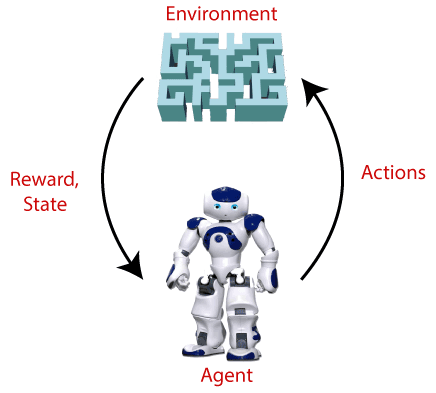
* Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labelled input data.
* Unsupervised learning is preferable as it is easy to get unlabelled data in comparison to labelled data.

## **Disadvantages of Unsupervised Learning**

* Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
* The result of the unsupervised learning algorithm might be less accurate as input data is not labelled, and algorithms do not know the exact output in advance.

1. **Reinforcement Learning**

* Reinforcement Learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favourable outputs are encouraged or ‘reinforced’, and non-favourable outputs are discouraged or ‘punished’.
* In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favourable or not.
* In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favourable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.
* In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm.
* Thus, the program is trained to give the best possible solution for the best possible reward.



## **Terms used in Reinforcement Learning**

* **Agent():** An entity that can perceive/explore the environment and act upon it.
* **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
* **Action():** Actions are the moves taken by an agent within the environment.
* **State():** State is a situation returned by the environment after each action taken by the agent.
* **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
* **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
* **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
* **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

## **Types of Reinforcement learning**

There are mainly two types of reinforcement learning, which are:

* **Positive Reinforcement**
* **Negative Reinforcement**

1. **Positive Reinforcement:**

The positive reinforcement learning means adding something to increase the tendency that expected behaviour would occur again. It impacts positively on the behaviour of the agent and increases the strength of the behaviour.

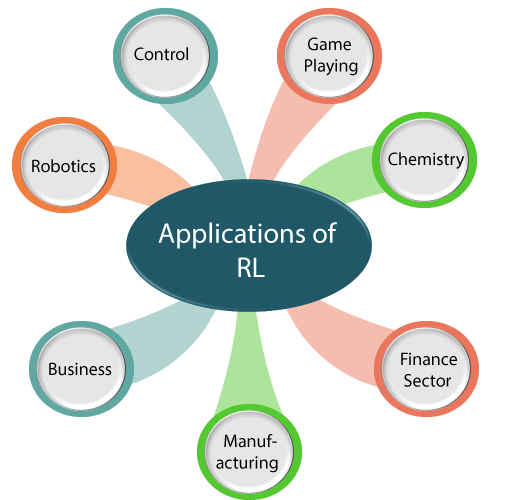
This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

1. **Negative Reinforcement:**

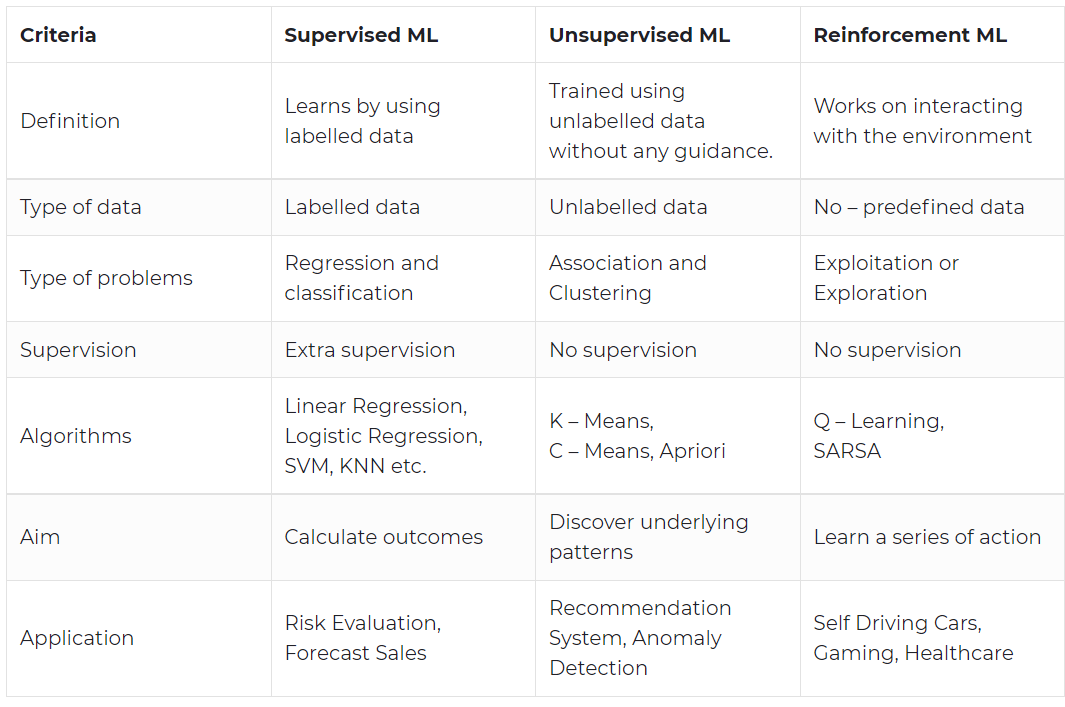
The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behaviour will occur again by avoiding the negative condition.

It can be more effective than the positive reinforcement depending on situation and behaviour, but it provides reinforcement only to meet minimum behaviour.

## **Reinforcement Learning Applications**

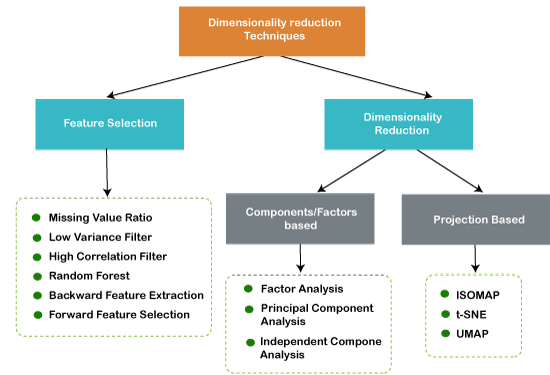


**Difference Between Supervised, Unsupervised and Reinforcement Learning**

****

## **4.3. DIMENSIONALITY REDUCTION**

* The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.
* A dataset contains a huge number of input features in various cases, which makes the predictive modelling task more complicated.
* Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.
* Dimensionality reduction technique can be defined as, **"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."**
* These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.
* It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc.**



## **The Curse of Dimensionality**

* Handling the high-dimensional data is very difficult in practice, commonly known as the curse of dimensionality.
* If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex.
* As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases.
* If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance.
* Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

## **Benefits of Dimensionality Reduction**

* By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
* Less Computation training time is required for reduced dimensions of features.
* Reduced dimensions of features of the dataset help in visualizing the data quickly.
* **I**t removes the redundant features (if present) by taking care of multicollinearity.

## **Disadvantages of dimensionality Reduction**

* Some data may be lost due to dimensionality reduction.
* In the PCA dimensionality reduction technique, sometimes the principal components required to consider are unknown.

## **4.3.1. Approaches of Dimension Reduction**

There are two ways to apply the dimension reduction technique, which are given below:

* Feature Selection
* Feature Extraction

### Feature Selection:

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset.

### Feature Extraction:

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

## **Common techniques of Dimensionality Reduction**

1. **Principal Component Analysis**
2. **Backward Elimination**
3. **Forward Selection**
4. **Score comparison**
5. **Missing Value Ratio**
6. **Low Variance Filter**
7. **High Correlation Filter**
8. **Random Forest**
9. **Factor Analysis**
10. **Auto-Encoder**

**4.4. Principal Component Analysis**

* Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.
* It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation.
* These new transformed features are called the **Principal Components**.
* It is one of the popular tools that is used for exploratory data analysis and predictive modelling.
* It is a technique to draw strong patterns from the given dataset by reducing the variances.
* PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.
* PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality.
* Example :  **image processing, movie recommendation system, etc.**

### Principal Components in PCA

* The transformed new features or the output of PCA are the Principal Components.
* The number of these PCs are either equal to or less than the original features present in the dataset.

Some properties of these principal components are given below:

* The principal component must be the linear combination of the original features.
* These components are orthogonal, i.e., the correlation between a pair of variables is zero.
* The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

**Algorithm of PCA**

The PCA algorithm is based on some mathematical concepts such as:

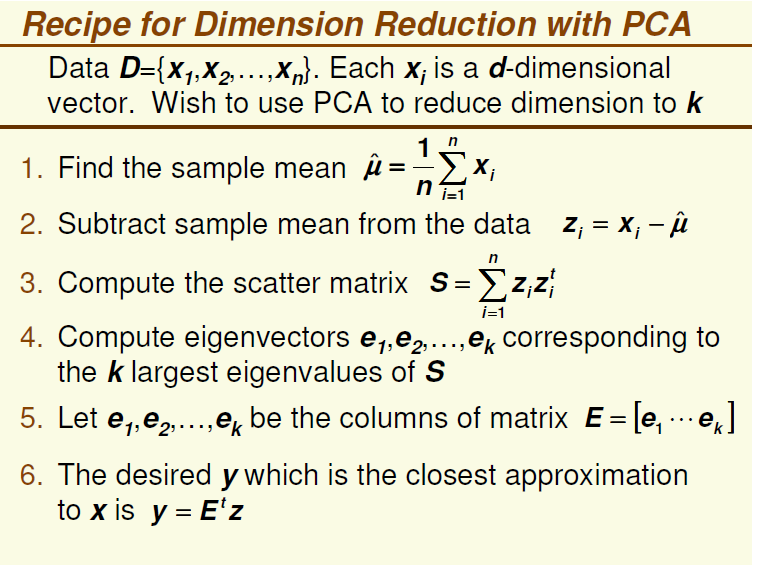
* Variance and Covariance
* Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

* **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
* **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
* **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
* **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
* **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

### Steps for PCA algorithm

1. **Getting the dataset**  
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**  
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**  
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance. If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**  
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**  
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**  
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P\*.
7. **Calculating the new features Or Principal Components**  
   Here we will calculate the new features. To do this, we will multiply the P\* matrix to the Z. In the resultant matrix Z\*, each observation is the linear combination of original features. Each column of the Z\* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**  
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.



### ****PCA Algorithm :****

The steps involved in PCA Algorithm are as follows-

* **Step-01:** Get data.
* **Step-02:** Compute the mean vector (µ).
* **Step-03:** Subtract mean from the given data.
* **Step-04:** Calculate the covariance matrix.
* **Step-05:** Calculate the eigen vectors and eigen values of the covariance matrix.
* **Step-06:** Choosing components and forming a feature vector.
* **Step-07:** Deriving the new data set.

## **PRACTICE PROBLEMS BASED ON PRINCIPAL COMPONENT ANALYSIS**

## **Problem-01:**

Given data = { 2, 3, 4, 5, 6, 7 }; {1, 5, 3, 6, 7, 8 }.

Compute the principal component using PCA Algorithm.

**OR**

Consider the two dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).

Compute the principal component using PCA Algorithm.

**OR**

Compute the principal component of following data-

**CLASS 1**

X = 2 , 3 , 4

Y = 1 , 5 , 3

**CLASS 2**

X = 5 , 6 , 7

Y = 6 , 7 , 8

## **Solution:**

### ****Step-01:****

Get data.

The given feature vectors are-

* x1 = (2, 1)
* x2 = (3, 5)
* x3 = (4, 3)
* x4 = (5, 6)
* x5 = (6, 7)
* x6 = (7, 8)

### ****Step-02:****

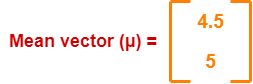
Calculate the mean vector (µ).

Mean vector (µ)

= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)

= (4.5, 5)

Thus,



**Step-03:**

Subtract mean vector (µ) from the given feature vectors.

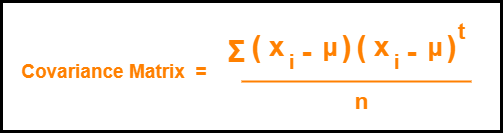
* x1 – µ, y1 - µ = (2 – 4.5, 1 – 5) = (-2.5, -4)
* x2 – µ, y2 - µ = (3 – 4.5, 5 – 5) = (-1.5, 0)
* x3 – µ, y3 - µ = (4 – 4.5, 3 – 5) = (-0.5, -2)
* x4 – µ, y4 - µ = (5 – 4.5, 6 – 5) = (0.5, 1)
* x5 – µ, y5 - µ = (6 – 4.5, 7 – 5) = (1.5, 2)
* x6 – µ, y6 - µ = (7 – 4.5, 8 – 5) = (2.5, 3)

Feature vectors (xi) after subtracting mean vector (µ) are-

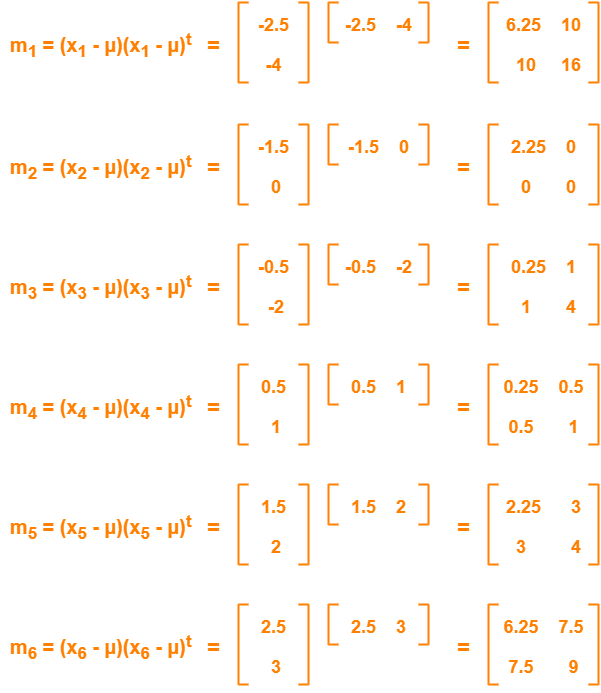


### ****Step-04:****

Calculate the covariance matrix.



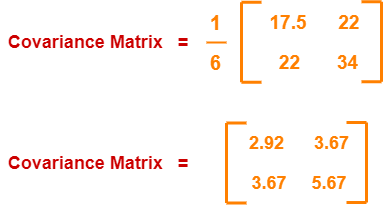
Hence,



Now,

**Covariance matrix = (m1 + m2 + m3 + m4 + m5 + m6) / 6**

On adding the above matrices and dividing by 6, we get-



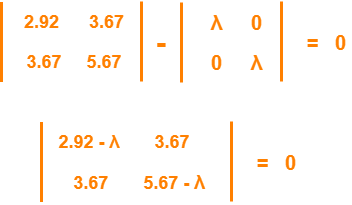
### ****Step-05:****

Calculate the eigen values and eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation

**|M – λI| = 0.**

So, we have-



From here,

(2.92 – λ)(5.67 – λ) – (3.67 x 3.67) = 0

16.56 – 2.92λ – 5.67λ + λ2 – 13.47 = 0

λ2 – 8.59λ + 3.09 = 0

Solving this quadratic equation, we get λ = 8.22, 0.38

Thus, two eigen values are λ1 = 8.22 and λ2 = 0.38.

Clearly, the second eigen value is very small compared to the first eigen value.

So, the second eigen vector can be left out.

Eigen vector corresponding to the greatest eigen value is the principal component for the given data set.

So. we find the eigen vector corresponding to eigen value λ1.

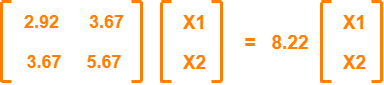
We use the following equation to find the eigen vector-

**MX = λX**

where-

* M = Covariance Matrix
* X = Eigen vector
* λ = Eigen value

Substituting the values in the above equation, we get-



Solving these, we get-

2.92X1 + 3.67X2 = 8.22X1

3.67X1 + 5.67X2 = 8.22X2

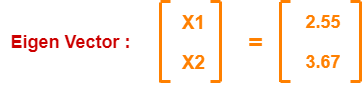
On simplification, we get-

5.3X1 = 3.67X2 ………(1)

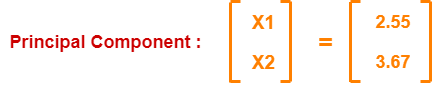
3.67X1 = 2.55X2 ………(2)

From (1) and (2), **X1 = 0.69X2**

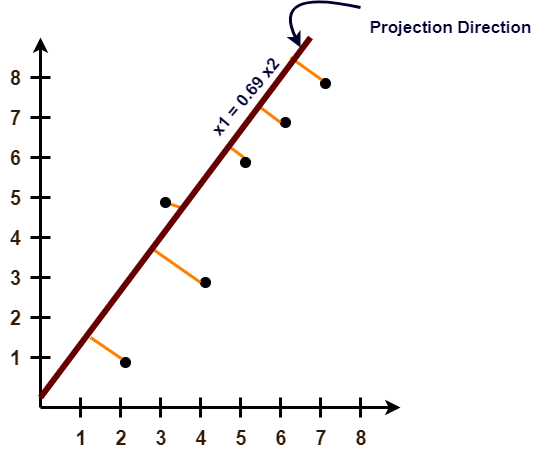
From (2), the eigen vector is-



Thus, principal component for the given data set is-



Lastly, we project the data points onto the new subspace as-



# **PCA Example using Python**

|  |
| --- |
| #Importing required libraries  import numpy as np |

### 1. Subtract the mean of each variable

Subtract the mean of each variable from the dataset so that the dataset should be centered on the origin. Doing this proves to be very helpful when calculating the covariance matrix.

|  |
| --- |
| #Generate a dummy dataset.  X **=** np.random.randint(10,50,100).reshape(20,5)  # mean Centering the data  X\_meaned **=** X **-** np.mean(X , axis **=** 0) |

Data generated by the above code have dimensions (20,5) i.e. 20 examples and 5 variables for each example. we calculated the mean of each variable and subtracted that from every row of the respective column.

### 2. Calculate the Covariance Matrix

Calculate the Covariance Matrix of the mean-centered data. The covariance matrix is a square matrix denoting the covariance of the elements with each other. The covariance of an element with itself is nothing but just its Variance.

So the diagonal elements of a covariance matrix are just the variance of the elements.

|  |
| --- |
| # calculating the covariance matrix of the mean-centered data.  cov\_mat **=** np.cov(X\_meaned , rowvar **=** False) |

We can find easily calculate covariance Matrix using numpy.cov( ) method. The default value for rowvar is set to True, remember to set it to False to get the covariance matrix in the required dimensions.

### 3. Compute the Eigenvalues and Eigenvectors

Now, compute the Eigenvalues and Eigenvectors for the calculated Covariance matrix. The Eigenvectors of the Covariance matrix we get are Orthogonal to each other and each vector represents a principal axis.

A Higher Eigenvalue corresponds to a higher variability. Hence the principal axis with the higher Eigenvalue will be an axis capturing higher variability in the data.

Orthogonal means the vectors are mutually perpendicular to each other.

|  |
| --- |
| #Calculating Eigenvalues and Eigenvectors of the covariance matrix  eigen\_values , eigen\_vectors **=** np.linalg.eigh(cov\_mat) |

NumPy linalg.eigh( ) method returns the eigenvalues and eigenvectors of a complex Hermitian or a real symmetric matrix.

### 4. Sort Eigenvalues in descending order

Sort the Eigenvalues in the descending order along with their corresponding Eigenvector.

Remember each column in the Eigen vector-matrix corresponds to a principal component, so arranging them in descending order of their Eigenvalue will automatically arrange the principal component in descending order of their variability.

Hence the first column in our rearranged Eigen vector-matrix will be a principal component that captures the highest variability.

**5. Select a subset from the rearranged Eigenvalue matrix**

Select a subset from the rearranged Eigenvalue matrix as per our need i.e. number\_comp = 2. This means we selected the first two principal components.

|  |
| --- |
| # select the first n eigenvectors, n is desired dimension of our final reduced data.    n\_components **=** 2 #you can select any number of components.  eigenvector\_subset **=** sorted\_eigenvectors[:,0:n\_components] |

n\_components = 2 means our final data should be reduced to just 2 variables. if we change it to 3 then we get our data reduced to 3 variables.

6. **Transform the data**

Finally, transform the data by having a dot product between the Transpose of the Eigenvector subset and the Transpose of the mean-centered data. By transposing the outcome of the dot product, the result we get is the data reduced to lower dimensions from higher dimensions.

|  |
| --- |
| #Transform the data  X\_reduced **=** np.dot(eigenvector\_subset.transpose(),X\_meaned.transpose()).transpose() |

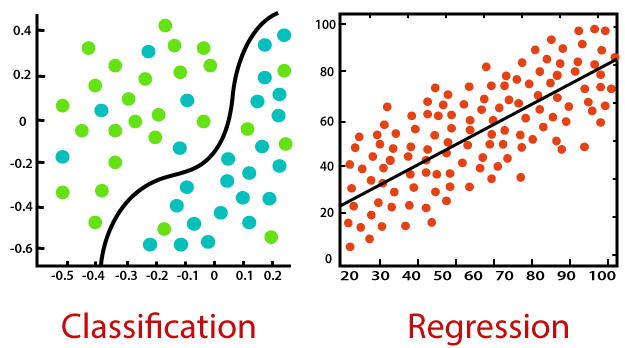
The final dimensions of X\_reduced will be ( 20, 2 ) and originally the data was of higher dimensions ( 20, 5 ).

# **4.5. REGRESSION AND CLASSIFICATION IN MACHINE LEARNING**

Regression and Classification algorithms are Supervised Learning algorithms. Both the algorithms are used for prediction in Machine learning and work with the labelled datasets.

The main difference between Regression and Classification algorithms that Regression algorithms are used to **predict the continuous** values such as price, salary, age, etc. and Classification algorithms are used to **predict/Classify the discrete values** such as Male or Female, True or False, Spam or Not Spam, etc.

Example :



## **4.5.1. REGRESSION**

Regression is a process of finding the correlations between dependent and independent variables. It helps in predicting the continuous variables such as prediction of **Market Trends**, prediction of House prices, etc.

The task of the Regression algorithm is to find the mapping function to map the input variable(x) to the continuous output variable(y).

**Example:** Suppose we want to do weather forecasting, so for this, we will use the Regression algorithm. In weather prediction, the model is trained on the past data, and once the training is completed, it can easily predict the weather for future days.

**Types of Regression Algorithm:**

* Simple Linear Regression
* Multiple Linear Regression
* Polynomial Regression
* Logistic Regression

# **Regression Analysis in Machine learning**

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc.

We can understand the concept of regression analysis using the below example:

**Example:** Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:

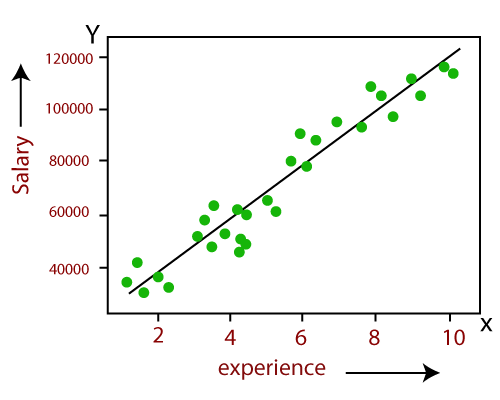
Now, the company wants to do the advertisement of $200 in the year 2019 **and wants to know the prediction about the sales for this year**. So to solve such type of prediction problems in machine learning, we need regression analysis.

## **Terminologies Related to the Regression Analysis:**

* **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.
* **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
* **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.
* **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
* **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

**Linear Regression:**

* Linear regression is a statistical regression method which is used for predictive analysis.
* It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.
* It is used for solving the regression problem in machine learning.
* Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.
* If there is only one input variable (x), then such linear regression is called **simple linear regression**.
* And if there is more than one input variable, then such linear regression is called **multiple linear regression**.
* The relationship between variables in the linear regression model can be explained using the below image.
* Here we are predicting the salary of an employee on the basis of **the year of experience**.



Below is the mathematical equation for Linear regression:

***Y= aX+b***

**Here,**

**Y =** dependent variables (target variables) **X =** Independent variables (predictor variables)  
**a** and **b** are the linear coefficients

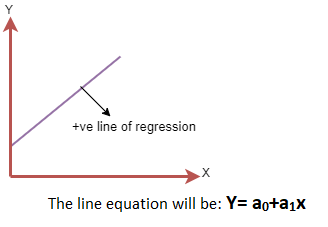
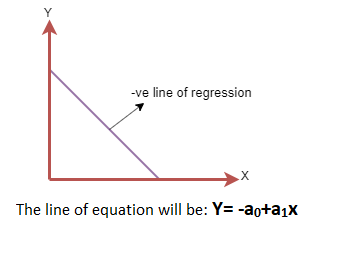
## **Types of Linear Regression :**

Linear regression can be further divided into two types of the algorithm:

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

## **Linear Regression Line**

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

* **Positive Linear Relationship:**  
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.
* **Negative Linear Relationship:**  
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

**Example of Linear Regression:**

Consider the following table with x,y:

|  |  |  |
| --- | --- | --- |
| **SUBJECT** | **AGE X** | **GLUCOSE LEVEL Y** |
| 1 | 43 | 99 |
| 2 | 21 | 65 |
| 3 | 25 | 79 |
| 4 | 42 | 75 |
| 5 | 57 | 87 |
| 6 | 59 | 81 |
| Σ | 247 | 486 |

**Step 1:** Calculate the following : xy, X2, Y2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SUBJECT** | **AGE X** | **GLUCOSE LEVEL Y** | **XY** | **X2** | **Y2** |
| 1 | 43 | 99 | 4257 | 1849 | 9801 |
| 2 | 21 | 65 | 1365 | 441 | 4225 |
| 3 | 25 | 79 | 1975 | 625 | 6241 |
| 4 | 42 | 75 | 3150 | 1764 | 5625 |
| 5 | 57 | 87 | 4959 | 3249 | 7569 |
| 6 | 59 | 81 | 4779 | 3481 | 6561 |
| **Σ** | **247** | **486** | **20485** | **11409** | **40022** |

From the above table,

Σx = 247,

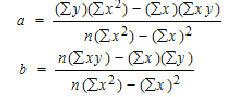
Σy = 486,

Σxy = 20485,

Σx2 = 11409,

Σy2 = 40022.

n is the sample size (6, in our case).

[](https://www.statisticshowto.com/wp-content/uploads/2009/11/linearregressionequations.bmp)**Step 2 :** Use the following equations to find a and b.

**Find a**:

= ((486 × 11,409) – ((247 × 20,485)) / 6 (11,409) – 2472)

= 484979 / 7445

= **65.14**

**Find b**:

= (6(20,485) – (247 × 486)) / (6 (11409) – 2472)

= (122,910 – 120,042) / 68,454 – 2472

= 2,868 / 7,445

= **0.385225**

**Step 3 :** Insert the values into the equation.

**y’ = a + bx**  
**y’ = 65.14 + .385225x**

Now, if suppose age is 34 then we can find the glucose level by substituting in the above equation:

**X=34**

**y’ = 65.14 + .385225x**

**y’ = 65.14 + .385225 x 34**

**y’ = 65.14 + 13.09765**

**y’ = 78.23765**

**Thus the glucose level for the person with age 34 is approximately 78.**

## **Linear regression using Python**

There are five basic steps when you’re implementing linear regression:

1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a regression model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions.

**Step 1: Import packages and classes**

The first step is to import the package numpy and the class LinearRegression from sklearn.linear\_model:

*import numpy as np*

*from sklearn.linear\_model import LinearRegression*

**Step 2: Provide data**

The inputs (regressors, 𝑥) and output (predictor, 𝑦) should be arrays (the instances of the class numpy.ndarray) or similar objects. This is the simplest way of providing data for regression:

*x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))*

*y = np.array([5, 20, 14, 32, 22, 38])*

The .reshape() on x variable is used because this array is required to be **two-dimensional**, or to be more precise, to have **one column and as many rows as necessary**. That’s exactly what the argument (-1, 1) of .reshape() specifies.

**Step 3: Create a model and fit it**

The next step is to create a linear regression model and fit it using the existing data.

Let’s create an instance of the class LinearRegression, which will represent the regression model:

*model = LinearRegression()*

This statement creates the variable model as the instance of LinearRegression. You can provide several optional parameters to LinearRegression:

* **fit\_intercept** is a boolean (True by default) that decides whether to calculate the intercept 𝑏₀ (True) or consider it equal to zero (False).
* **normalize** is a Boolean (False by default) that decides whether to normalize the input variables (True) or not (False).
* **copy\_X** is a Boolean (True by default) that decides whether to copy (True) or overwrite the input variables (False).
* **n\_jobs** is an integer or None (default) and represents the number of jobs used in parallel computation. None usually means one job and -1 to use all processors.

This example uses the default values of all parameters.

With .fit(), you calculate the optimal values of the weights 𝑏₀ and 𝑏₁, using the existing input and output (x and y) as the arguments. In other words, .fit() **fits the model**. It returns self, which is the variable model itself. That’s why you can replace the last two statements with this one:

*model = LinearRegression().fit(x, y)*

**Step 4: Get results**

Once you have your model fitted, you can get the results to check whether the model works satisfactorily and interpret it.

You can obtain the coefficient of determination (𝑅²) with .score() called on model:

*r\_sq = model.score(x, y)*

*print('coefficient of determination:', r\_sq)*

**coefficient of determination: 0.715875613747954**

When you’re applying .score(), the arguments are also the predictor x and regressor y, and the return value is 𝑅².

The attributes of model are .intercept\_, which represents the coefficient, 𝑏₀ and .coef\_, which represents 𝑏₁:

*print('intercept:', model.intercept\_)*

**intercept: 5.633333333333329**

*print('slope:', model.coef\_)*

**slope: [0.54]**

The code above illustrates how to get 𝑏₀ and 𝑏₁. You can notice that .intercept\_ is a scalar, while .coef\_ is an array.

The value 𝑏₀ = 5.63 (approximately) illustrates that your model predicts the response 5.63 when 𝑥 is zero. The value 𝑏₁ = 0.54 means that the predicted response rises by 0.54 when 𝑥 is increased by one.

**Step 5: Predict response**

Once there is a satisfactory model, you can use it for predictions with either existing or new data.

To obtain the predicted response, use .predict():

*y\_pred = model.predict(x)*

*print('predicted response:', y\_pred)*

**predicted response:**

**[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]**

When applying .predict(), you pass the regressor as the argument and get the corresponding predicted response.

**4.5.2. CLASSIFICATION**

Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes.

The task of the classification algorithm is to find the mapping function to map the input(x) to the discrete output(y).

**Example:** The best example to understand the Classification problem is Email Spam Detection. The model is trained on the basis of millions of emails on different parameters, and whenever it receives a new email, it identifies whether the email is spam or not. If the email is spam, then it is moved to the Spam folder.

**Types of ML Classification Algorithms:**

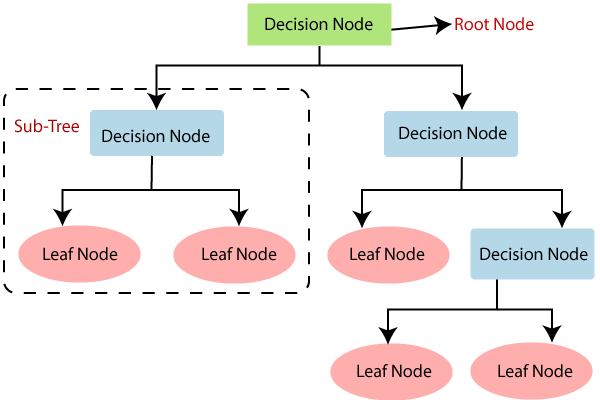
Classification Algorithms can be further divided into the following types:

* K-Nearest Neighbours
* Support Vector Machines
* Kernel SVM
* Naïve Bayes
* Decision Tree Classification
* Random Forest Classification

## **Difference between Regression and Classification :**

|  |  |
| --- | --- |
| **Regression Algorithm** | **Classification Algorithm** |
| In Regression, the output variable must be of continuous nature or real value. | In Classification, the output variable must be a discrete value. |
| The task of the regression algorithm is to map the input value (x) with the continuous output variable(y). | The task of the classification algorithm is to map the input value(x) with the discrete output variable(y). |
| Regression Algorithms are used with continuous data. | Classification Algorithms are used with discrete data. |
| In Regression, we try to find the best fit line, which can predict the output more accurately. | In Classification, we try to find the decision boundary, which can divide the dataset into different classes. |
| Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc. | Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc. |
| The regression Algorithm can be further divided into Linear and Non-linear Regression. | The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier. |

# **DECISION TREE CLASSIFICATION ALGORITHM**

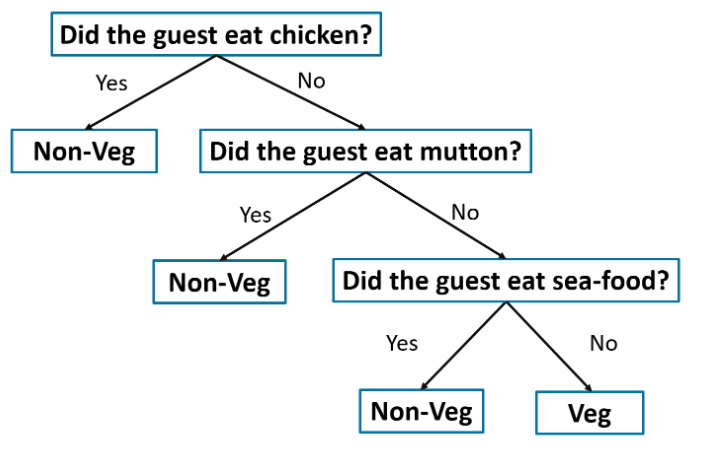
* Decision Tree is a **Supervised learning technique**that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where**internal nodes represent the features of a dataset, branches represent the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

**Examples:**

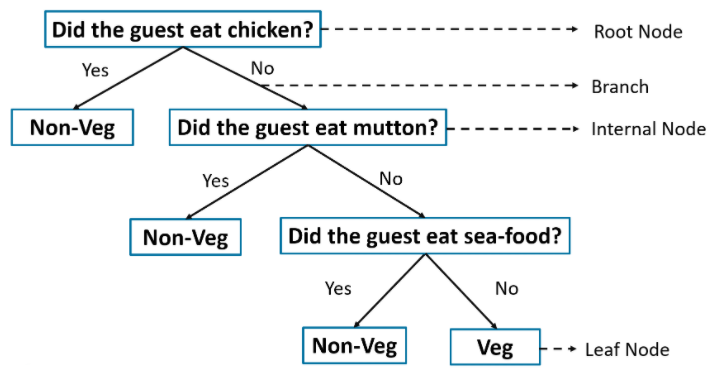
1. Predicting an email as spam or not spam
2. Predicting of a tumor is cancerous
3. Predicting a loan as a good or bad credit risk based on the factors in each of these.

Generally, a model is created with observed data also called **training data**. Then a set of **validation data** is used to *verify* and improve the model.

Suppose you hosted a huge party and you want to know how many of your guests were non-vegetarians. To solve this problem, a simple Decision Tree is used.



**Structure of Decision Tree :**



## **Decision Tree Terminologies**

* **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
* **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
* **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
* **Branch/Sub Tree:** A tree formed by splitting the tree.
* **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
* **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

**Decision Tree Algorithm :**

The Decision Tree Algorithm follows the below steps:

**Step 1:** Select the feature (predictor variable) that best classifies the data set into the desired classes and assign that feature to the root node.

**Step 2:** Traverse down from the root node, while making relevant decisions at each internal node such that each internal node best classifies the data.

**Step 3:** Route back to step 1 and repeat until you assign a class to the input data.

**Types of Decision Tree Algorithm:**

* **Decision stump**  
  Used for generating a decision tree with just a single split hence also known as a one-level decision tree. It is known for its low predictive performance in most cases due to its simplicity.
* **M5**  
  Known for its precise classification accuracy and its ability to work well to a boosted decision tree and small datasets with too much noise.
* **ID3(Iterative Dichotomiser 3)**  
  One of the *core and widely used decision tree algorithms* uses a top-down, greedy search approach through the given dataset and selects the best attribute for classifying the given dataset
* **C4.5**  
  Also known as the statistical classifier this type of decision tree is derived from its parent ID3. This generates decisions based on a bunch of predictors.
* **C5.0**  
  Being the successor of the C4.5 it broadly has two models namely the basic tree and rule-based model, and its nodes can only predict categorical targets.
* **CHAID**  
  Expanded as Chi-squared Automatic Interaction Detector, this algorithm basically studies the merging variables to justify the outcome on the dependant variable by structuring a predictive model
* **MARS**  
  Expanded as multivariate adaptive regression splines, this algorithm creates a series of piecewise linear models which is used to model irregularities and interactions among variables, they are known for their ability to handle numerical data with greater efficiency.
* **Conditional Inference Trees**  
  This is a type of decision tree that uses a conditional inference framework to recursively segregate the response variables, it’s known for its flexibility and strong foundations.
* **CART**  
  Expanded as Classification and Regression Trees, the values of the target variables are predicted if they are continuous else the necessary classes are identified if they are categorical.

**Decision tree using id3 algorithm :**

* ID3 or the Iterative Dichotomiser 3 algorithm is one of the most effective algorithms used to build a Decision Tree.
* It uses the concept of ***Entropy*** and ***Information Gain*** to generate a Decision Tree for a given set of data.

**ID3 algorithm**

* The ID3 algorithm follows the below workflow in order to build a Decision Tree:
  + Select **Best Attribute** (A)
  + Assign A as a decision variable for the root node.
  + For each value of A, build a descendant of the node.
  + Assign classification labels to the leaf node.
  + If data is correctly classified: Stop.
  + Else: Iterate over the tree.

**1. Entropy**

* Entropy measures the impurity or uncertainty present in the data. It is used to decide how a Decision Tree can split the data.
* If the sample is completely uniform then entropy is 0, if it’s uniformly partitioned it is one. Higher the entropy more difficult it becomes to draw conclusions from that information.
* **Equation For Entropy:**

**2. Information Gain (IG)**

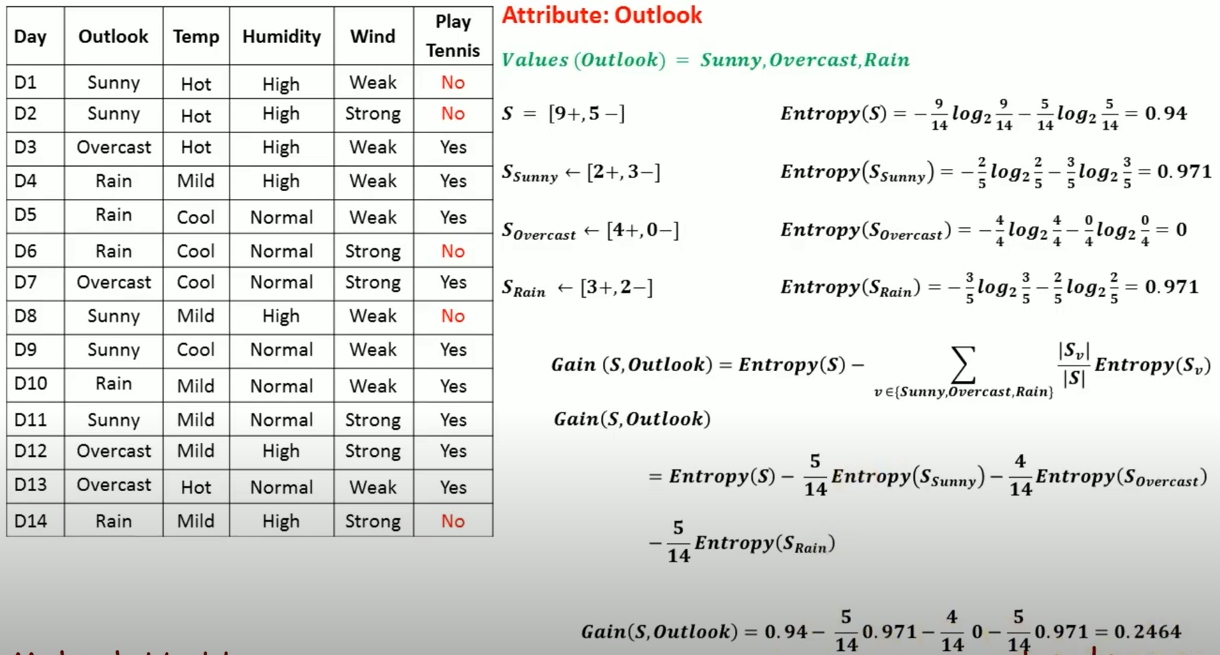
* Information Gain (IG) is the most significant measure used to build a Decision Tree.
* It indicates how much “information” a particular feature/ variable gives us about the final outcome.
* Information Gain is important because it used to choose the variable that best splits the data at each node of a Decision Tree.
* The variable with the highest IG is used to split the data at the root node.
* **Equation For Information Gain (IG):**

***Example of Decision Tree using ID3 Algorithm:***

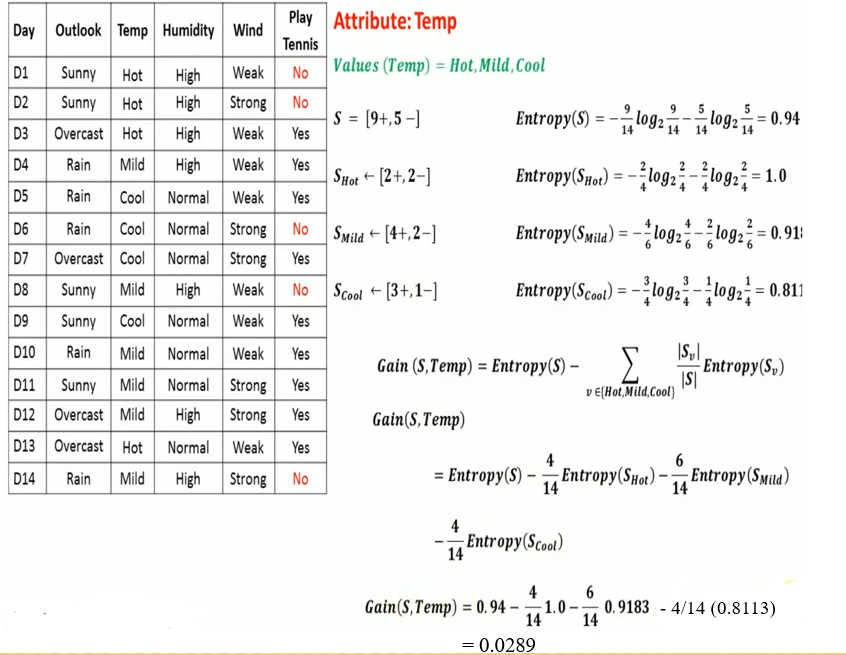
At which days the children's will play Tennis?

****

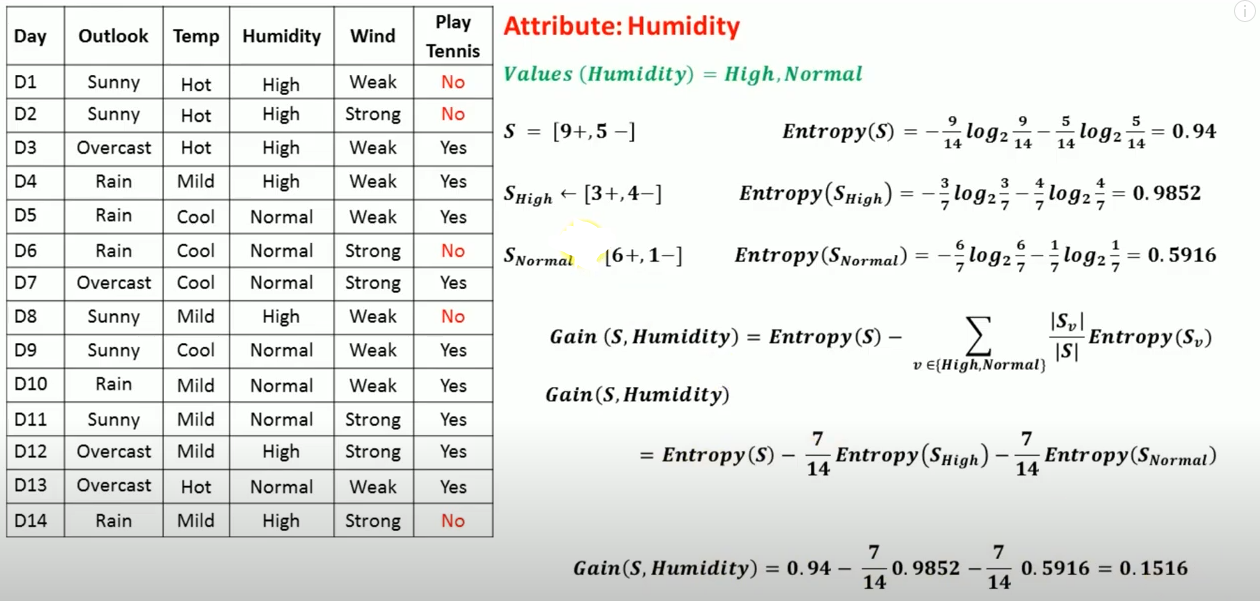
**Step :1**



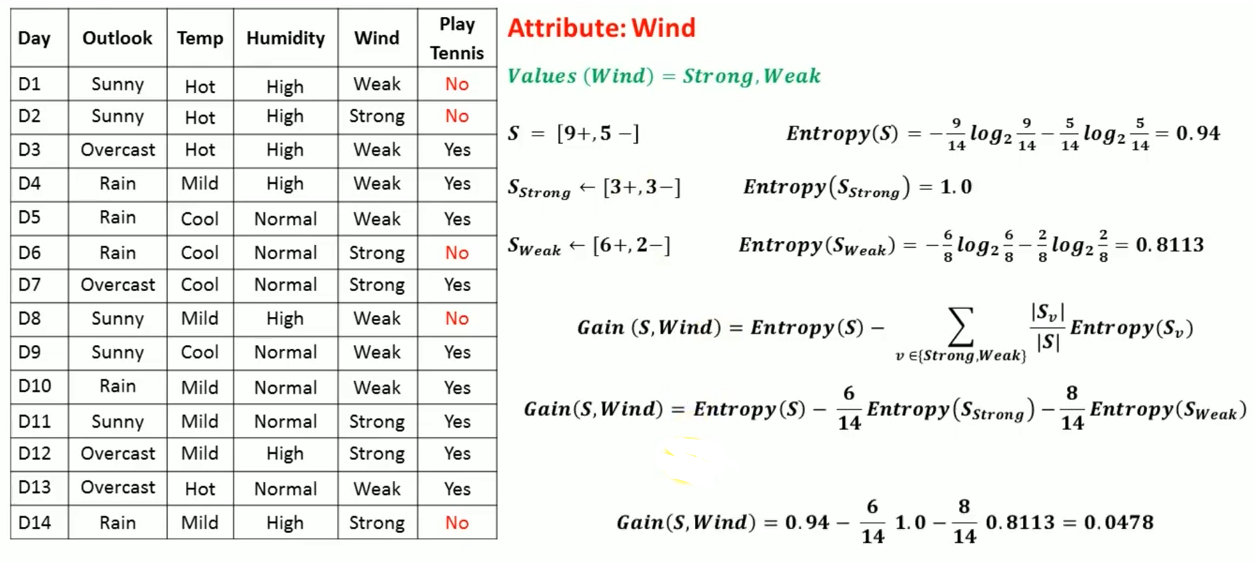
**Step 2:**



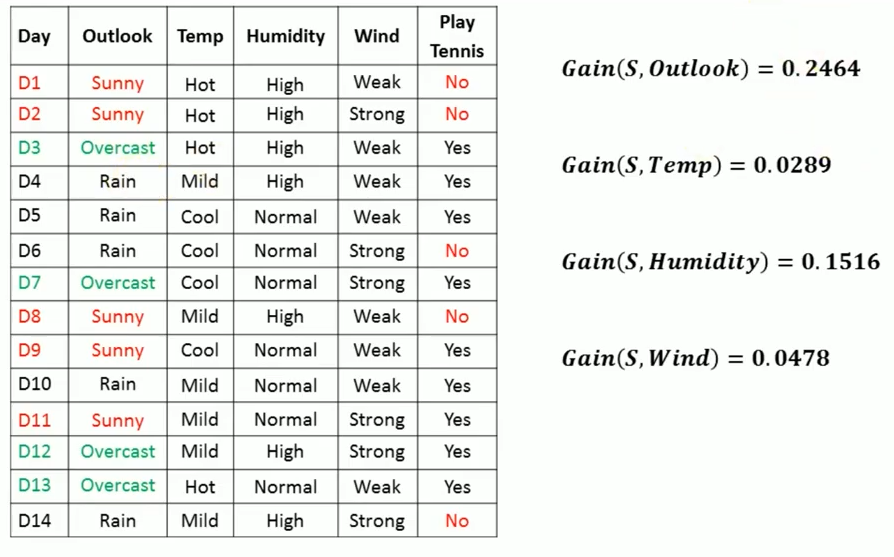
**Step 3:**

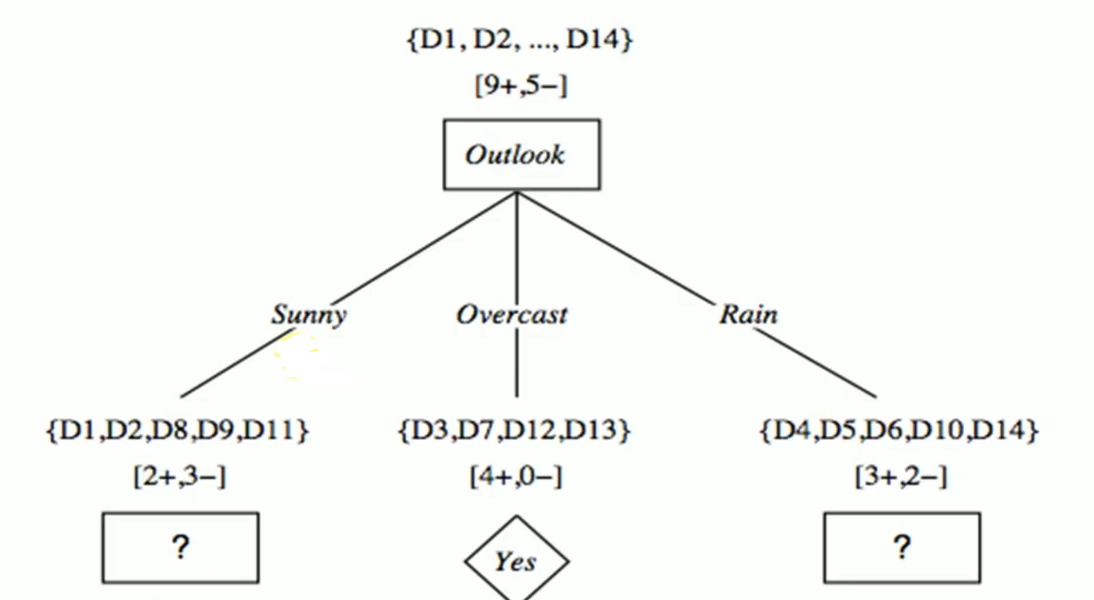


**Step 4:**

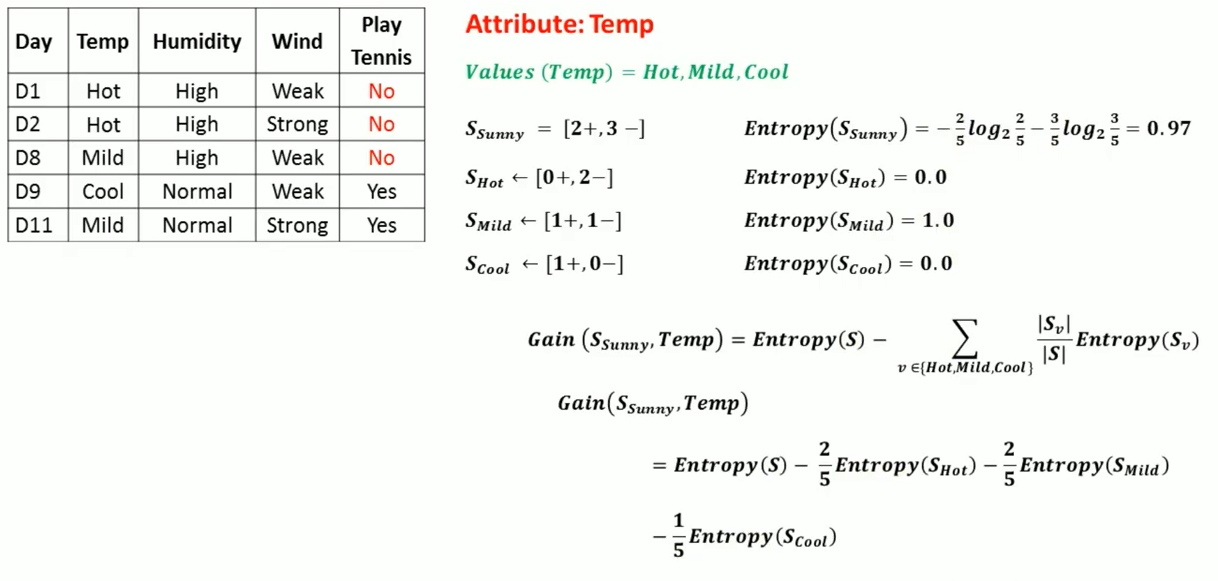
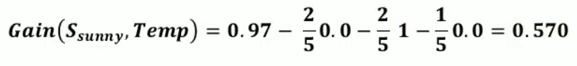


**Step 5:** Choose the maximum gain…

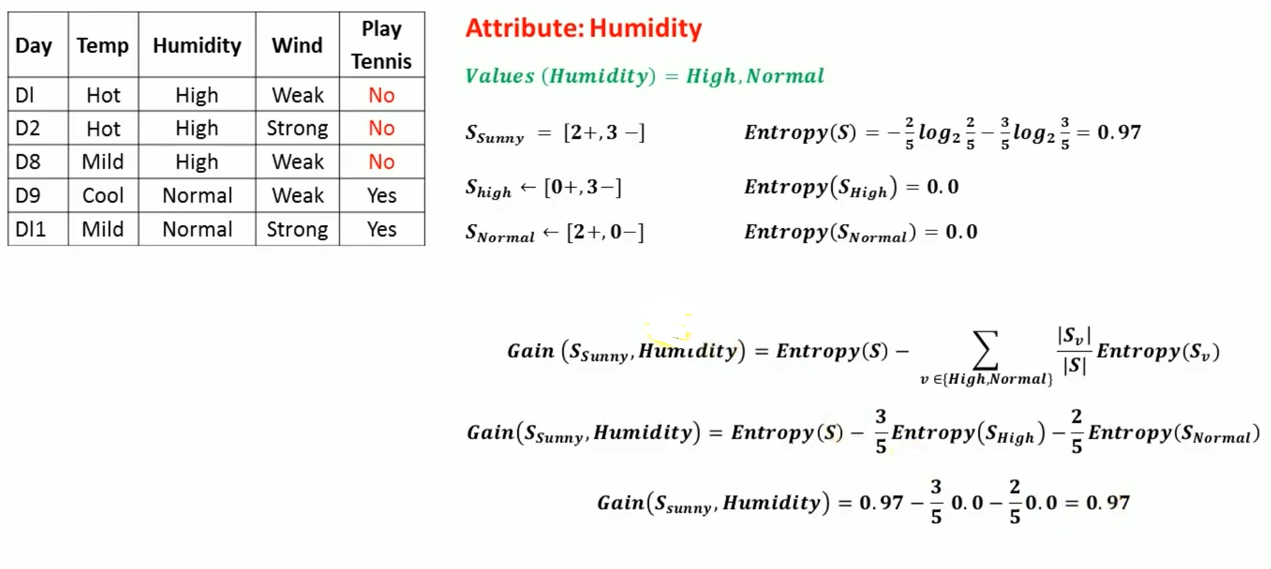




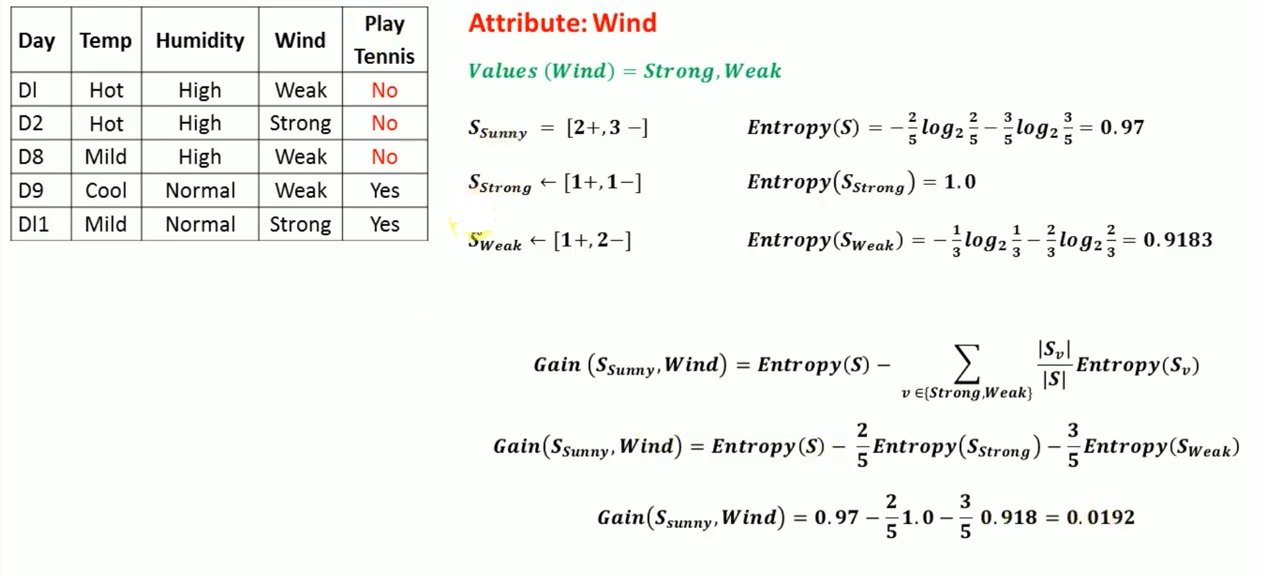
**Step 6:**

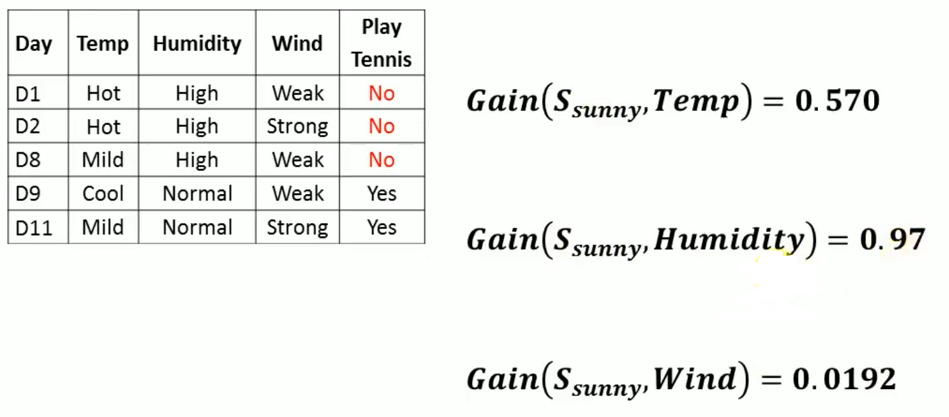
**Step 7:**

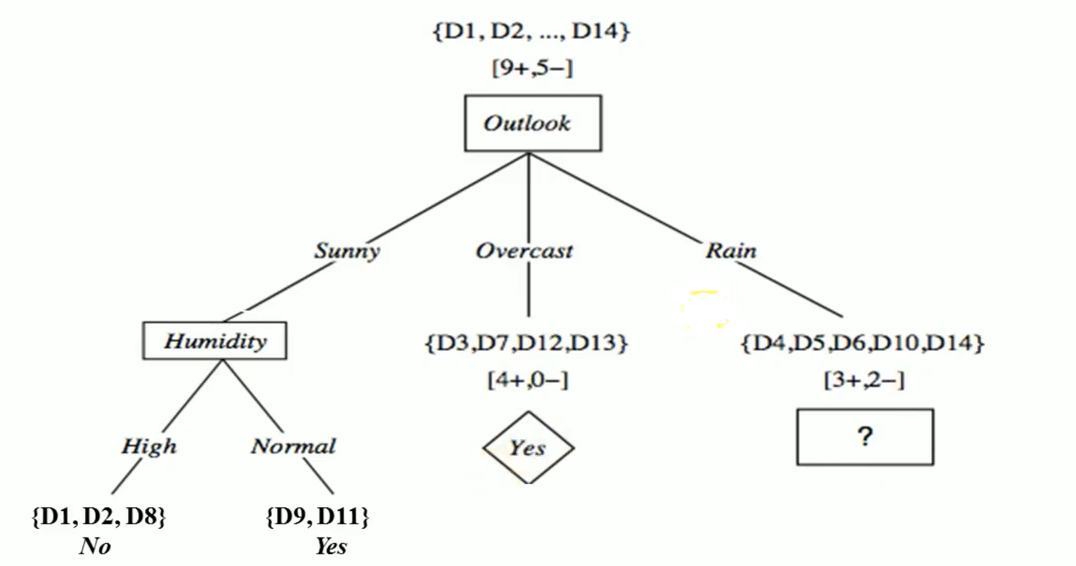


**Step 8:**

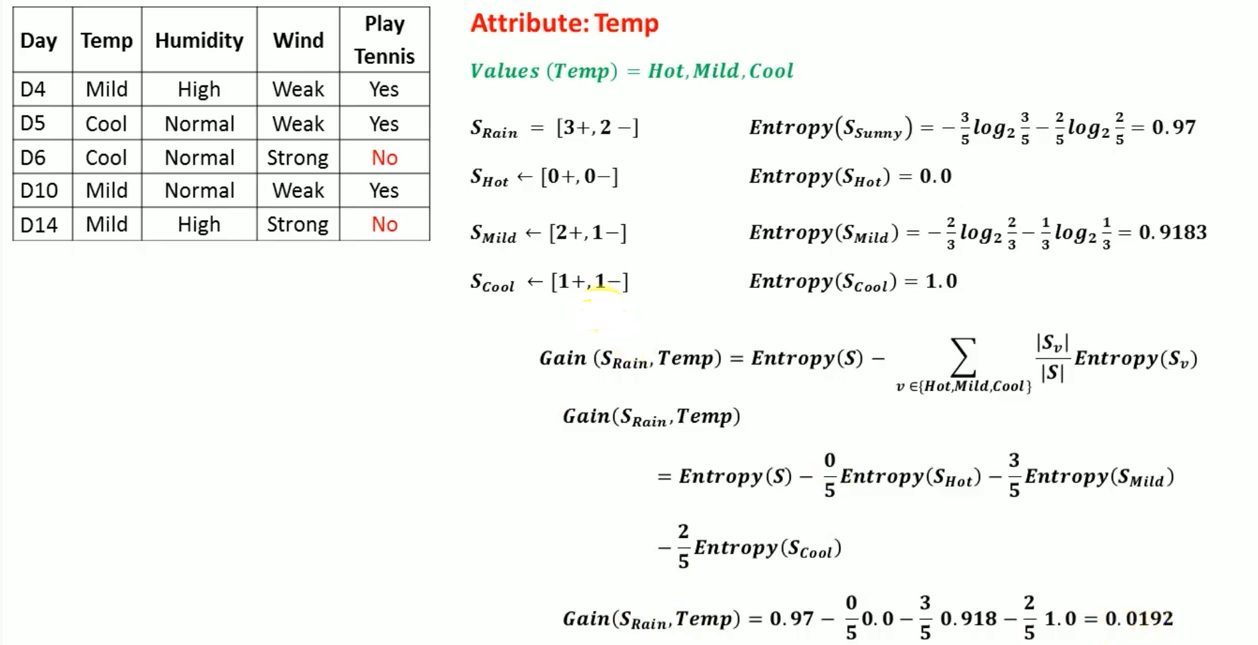


**Step 9:** Choose the maximum gain…

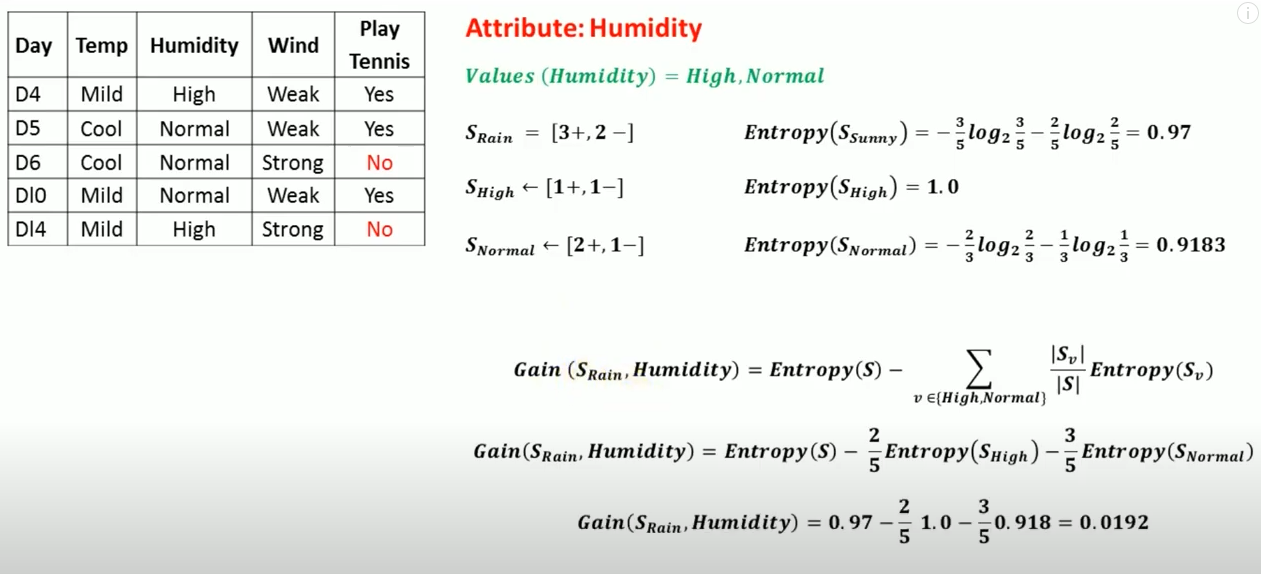




**Step 10:**



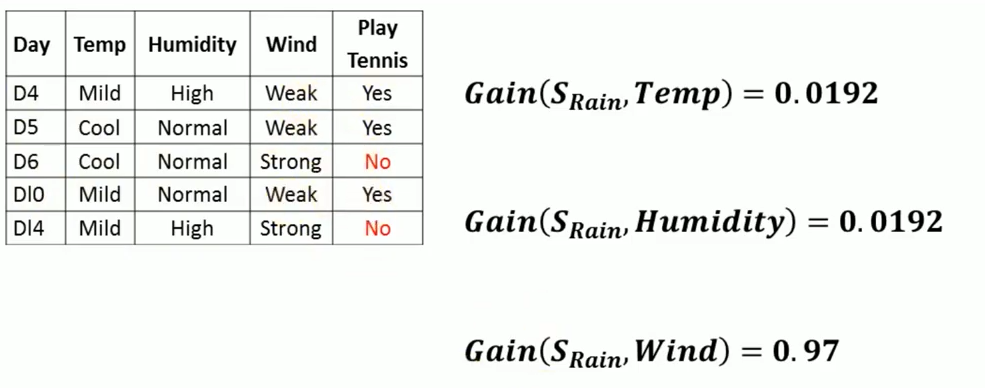
**Step 11:**

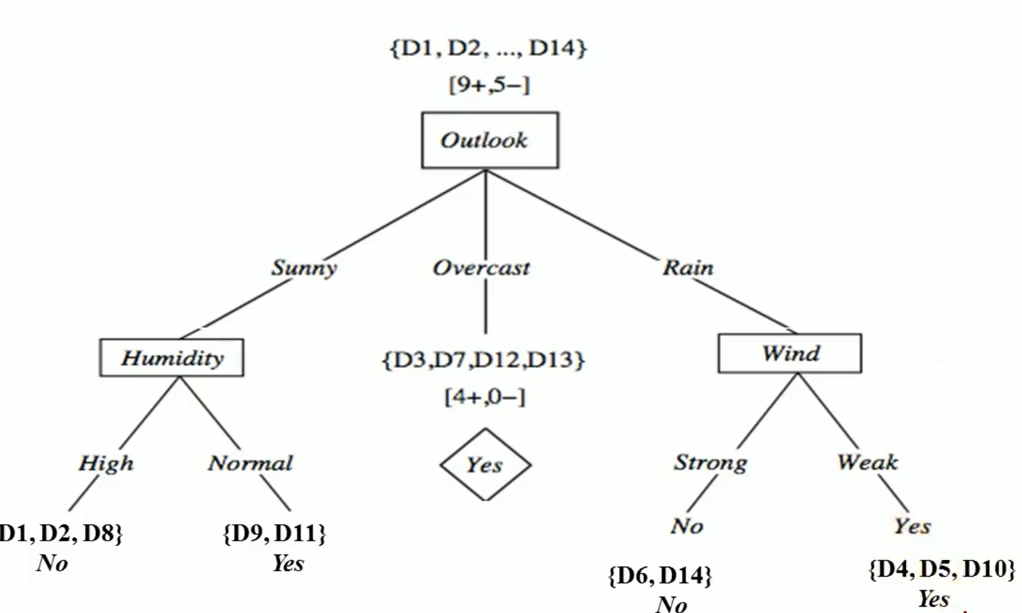


**Step 12:**



**Step 13:** Choose the maximum gain…



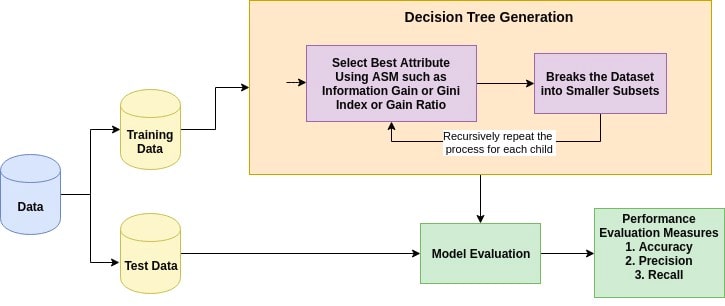


**Advantages**

* Easy to understand and interpret.
* Does not require Data normalization
* Doesn’t facilitate the need for scaling of data
* The preprocessing stage requires lesser effort compared to other major algorithms, hence in a way optimizes the given problem

**Disadvantages**

* Requires higher time to train the model
* It has a considerable high complexity and takes more time to process the data
* When decrease in user input parameter is very small it leads to the termination of the tree
* Calculations can get very complex at times



**Decision Tree using Python :** (Dummy Data)

*from sklearn import tree*

*X = [[0, 0], [1, 1], [0, 1], [1, 0], [2, 2]]*

*Y = [0, 1, 0, 1, 2]*

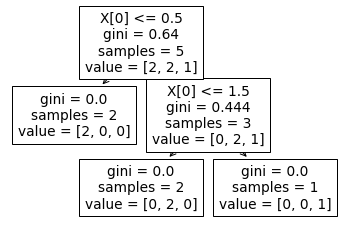
*clf = tree.DecisionTreeClassifier()*

*clf = clf.fit(X, Y)*

*print(clf.predict([[2., 1.]]))*

Now, plot using the built-in plot\_tree in the tree module

*tree.plot\_tree(clf)*



At the root node, we have 5 samples. It checks for the first value in X and if it's less than or equal to 0.5, it classifies the sample as **0.**If it’s not, then it checks if the first value in X is less than or equal to 1.5, in which case it assigns the label **1**to it, otherwise **2.**

Note that the decision tree doesn’t include any check on the second value in X. This is not an error as the second value is not needed in this case. If the decision tree is able to make all the classifications without the need for all the features, then it can ignore other features.

# **4.6. BAYESIAN 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**.

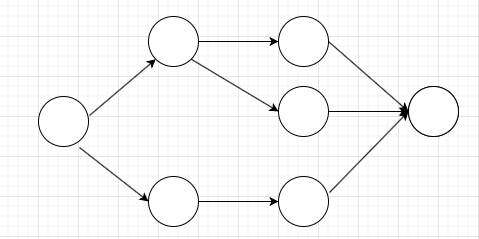
Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

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**.

## **What is a Bayesian Network?**

A Bayesian network falls under the category of Probabilistic Graphical Modelling technique, which is used to calculate uncertainties by using the notion of probability.

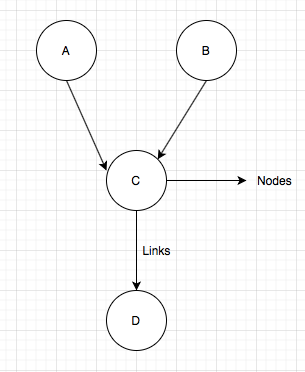
They are used to model improbability using directed acyclic graphs.



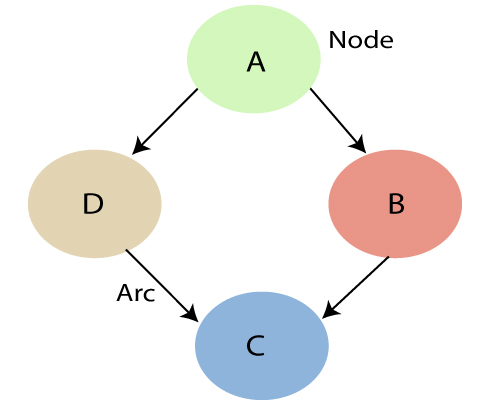
Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

* Directed Acyclic Graph
* Table of conditional probabilities.

## **What is Directed Acyclic Graph?**

It is used to represent the Bayesian Network. A directed acyclic graph contains nodes and links, where links denote the relationship between nodes.

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(Xi |Parent(Xi) )**, which determines the effect of the parent on that node.

Bayesian network is based on *Joint probability distribution* and *conditional probability.*

## **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[x1, x2, x3,....., xn]**, it can be written as the following way in terms of the joint probability distribution.

**= P[x1| x2, x3,....., xn]P[x2, x3,....., xn]**

**= P[x1| x2, x3,....., xn]P[x2|x3,....., xn]....P[xn-1|xn]P[xn].**

In general for each variable Xi, we can write the equation as:

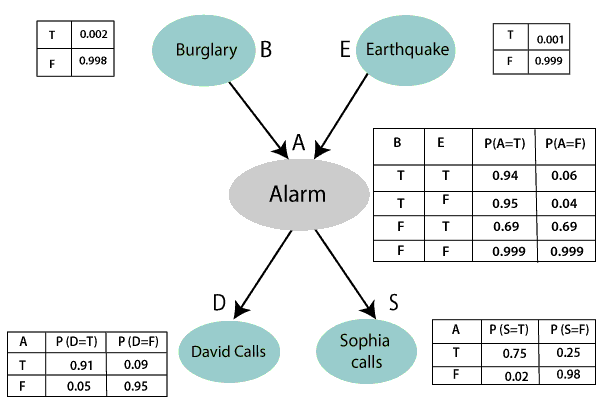
P(Xi|Xi-1,........., X1) = P(Xi |Parents(Xi ))

**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 neighbours 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.**

**Solution:**

* The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
* The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
* The conditional distributions for each node are given as conditional probabilities table or CPT.
* Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
* In CPT, a boolean variable with k boolean parents contains 2K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

**List of all events occurring in this network:**

* Burglary (B)
* Earthquake(E)
* Alarm(A)
* David Calls(D)
* Sophia calls(S)

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

**P[D, S, A, B, E] = P[D | S, A, B, E]. P[S, A, B, E]**

**= P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]**

**= P [D| A]. P [ S| A, B, E]. P[ A, B, E]**

**= P[D | A]. P[ S | A]. P[A| B, E]. P[B, E]**

**= P[D | A ]. P[S | A]. P[A| B, E]. P[B |E]. P[E]**

Let's take the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary.

P(B= False)= 0.998, which is the probability of no burglary.

P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

**Conditional probability table for Alarm A:**

The Conditional probability of Alarm A depends on Burglar and earthquake:

|  |  |  |  |
| --- | --- | --- | --- |
| **B** | **E** | **P(A= True)** | **P(A= False)** |
| True | True | 0.94 | 0.06 |
| True | False | 0.95 | 0.04 |
| False | True | 0.31 | 0.69 |
| False | False | 0.001 | 0.999 |

**Conditional probability table for David Calls:**

The Conditional probability of David that he will call depends on the probability of Alarm.

|  |  |  |
| --- | --- | --- |
| **A** | **P(D= True)** | **P(D= False)** |
| True | 0.91 | 0.09 |
| False | 0.05 | 0.95 |

**Conditional probability table for Sophia Calls:**

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

|  |  |  |
| --- | --- | --- |
| **A** | **P(S= True)** | **P(S= False)** |
| True | 0.75 | 0.25 |
| False | 0.02 | 0.98 |

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

**P(S, D, A, ¬B, ¬E) =** P (S|A) \*P (D|A)\*P (A|¬B ^ ¬E) \*P (¬B) \*P (¬E)

= 0.75\* 0.91\* 0.001\* 0.998\*0.999

**=** 0.00068045.

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

## **Bayesian Networks in Python**

Bayesian Networks can be developed and used for inference in Python.

A popular library for this is called **PyMC** and provides a range of tools for Bayesian modelling, including graphical models like Bayesian Networks.

The most recent version of the library is called **PyMC3**, named for Python version 3, and was developed on top of the Theano mathematical computation library that offers fast automatic differentiation.

**Installation with environment:**

*conda create -n BNLEARN python=3.6*

*conda activate BNLEARN*

*conda install -c ankurankan pgmpy*

*conda deactivate*

*conda activate BNLEARN*

*pip install bnlearn*

# Load titanic dataset containing mixed variables

*df\_raw = bnlearn.import\_example(data='titanic'****)***

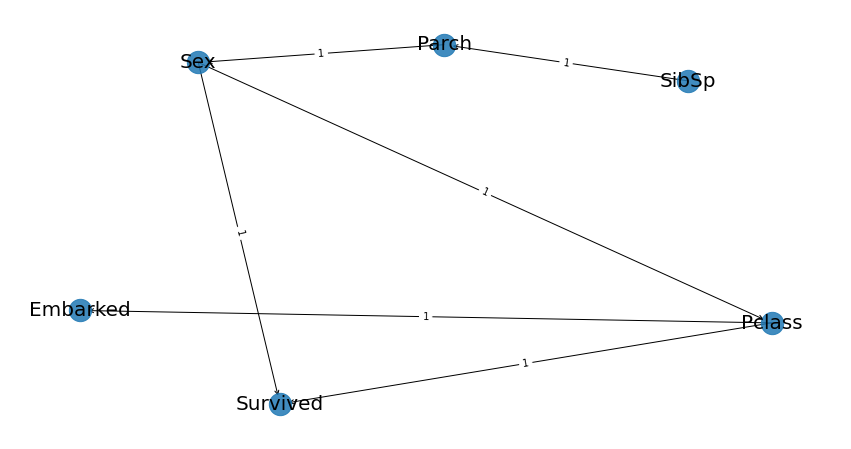
# Pre-processing of the input dataset

*dfhot, dfnum = bnlearn.df2onehot(df\_raw)*

# Structure learning

*DAG = bnlearn.structure\_learning.fit(dfnum)*

# Plot

[](https://i.stack.imgur.com/HbNdu.png)*G = bnlearn.plot(DAG)*

# Parameter learning

*model = bnlearn.parameter\_learning.fit(DAG, df)*

# Print CPDs

*bnlearn.print\_CPD(model)*

# Make inference

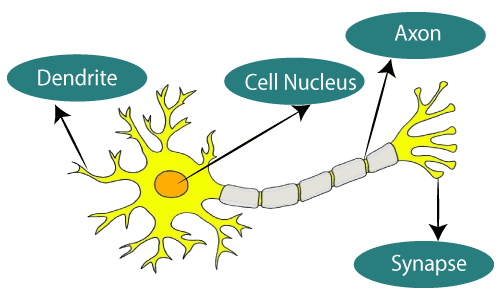
*q = bnlearn.inference.fit(model, variables=['Survived'], evidence={'Sex':0, 'Pclass':1})*

*print(q.values)*

*print(q.variables)*

**4.7. NEURAL NETWORK**

Neural Networks is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses.

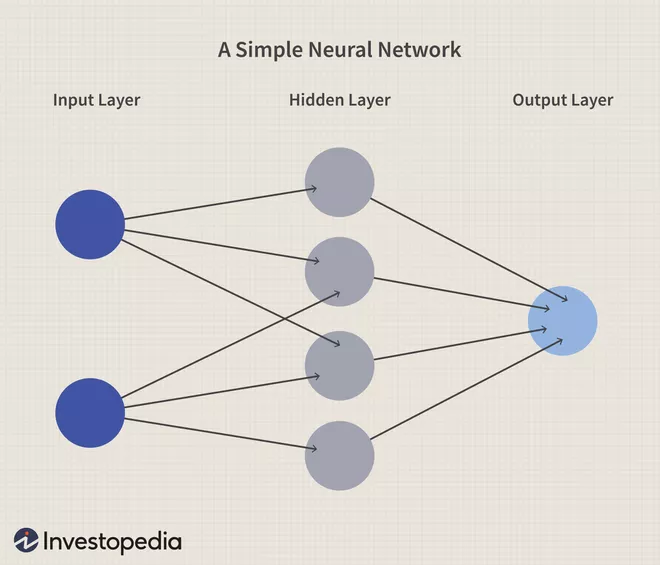
In simple words, Neural Networks are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain/biology.

**The figure illustrates the typical diagram of Biological Neural Network.**

### Components / Architecture of Neural Network

A simple neural network consists of three components :

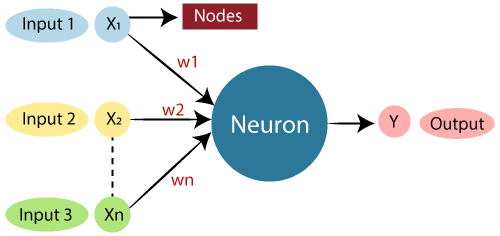
* Input layer
* Hidden layer
* Output layer



**Input Layer:**Also known as Input nodes are the inputs/information from the outside world is provided to the model to learn and derive conclusions from. Input nodes pass the information to the next layer i.e Hidden layer.

**Hidden Layer:** Hidden layer is the set of neurons where all the computations are performed on the input data. There can be any number of hidden layers in a neural network. The simplest network consists of a single hidden layer.

**Output layer:**The output layer is the output/conclusions of the model derived from all the computations performed. There can be single or multiple nodes in the output layer. If we have a binary classification problem the output node is 1 but in the case of multi-class classification, the output nodes can be more than 1.



The typical Artificial Neural Network looks something like the above figure.

**Relationship between Biological neural network and artificial neural network:**

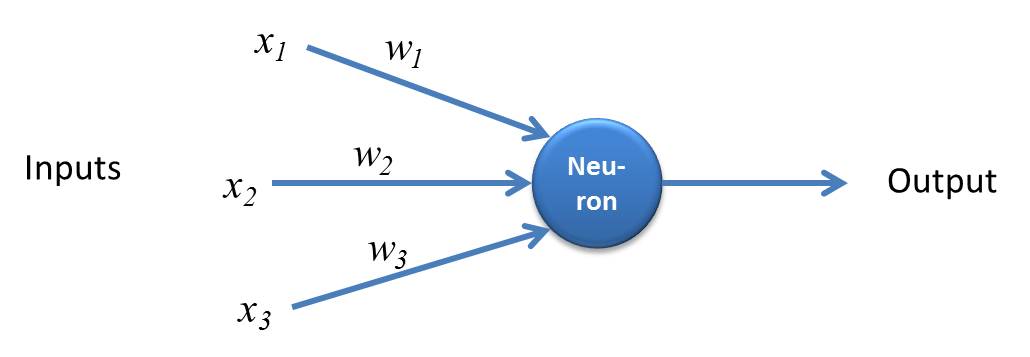
|  |  |
| --- | --- |
| **Biological Neural Network** | **Artificial Neural Network (ANN)** |
| Dendrites | Inputs |
| Cell nucleus | Nodes |
| Synapse | Weights |
| Axon | Output |

What is Artificial Neural NetworkThe artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

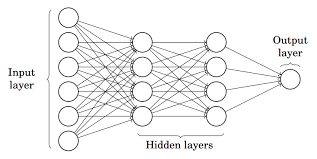
It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

## **Perceptron and Multi-Layer Perceptron**

**Perceptron** is a simple form of Neural Network and consists of a single layer where all the mathematical computations are performed.



Whereas, **Multilayer Perceptron** also known as **Artificial Neural Networks** consists of more than one perception which is grouped together to form a multiple layer neural network.



In the above figure, The Artificial Neural Network consists of four layers interconnected with each other:

* An input layer, with 6 input nodes
* Hidden Layer 1, with 4 hidden nodes/4 perceptrons
* Hidden layer 2, with 4 hidden nodes
* Output layer with 1 output node

## **Step by Step Working of the Artificial Neural Network**

## Step by Step Working of the artificial Neural Network

1. In the first step, **Input units are passed i.e data is passed with some weights attached to it to the hidden layer**. We can have any number of hidden layers. In the above image inputs x1,x2,x3,….xn is passed.
2. Each hidden layer consists of neurons. All the inputs are connected to each neuron.
3. After passing on the inputs,**all the computation is performed in the hidden layer**.

Computation performed in hidden layers are done in two steps which are as follows :

* First of all, **all the inputs are multiplied by their weights**. Weight is the gradient or coefficient of each variable. It shows the strength of the particular input. After assigning the weights, a bias variable is added. **Bias** is a constant that helps the model to fit in the best way possible.

***Z1 = W1\*In1 + W2\*In2 + W3\*In3 + W4\*In4 + W5\*In5 + b***

W1, W2, W3, W4, W5 are the weights assigned to the inputs In1, In2, In3, In4, In5, and b is the bias.

* Then in the second step, the**activation function is applied to the linear equation Z1.** The activation function is a non-linear transformation that is applied to the input before sending it to the next layer of neurons. The importance of the activation function is to inculcate nonlinearity in the model.

1. The whole process described in point 3 is performed in each hidden layer. After passing through every hidden layer, **we move to the last layer i.e our output layer which gives us the final output.**

The process explained above is known as **forwarding Propagation.**

1. After getting the predictions from the output layer, the **error is calculated i.e the difference between the actual and the predicted output.**

If the error is large, then the steps are taken to minimize the error and for the same purpose, **Back Propagation is performed.**

## **What is Back Propagation and How it works?**

Back Propagation is the process of updating and finding the optimal values of weights or coefficients which helps the model to minimize the error i.e difference between the actual and predicted values.

**The weights are updated with the help of optimizers**. Optimizers are the methods/ mathematical formulations to change the attributes of neural networks i.e weights to minimize the error.

## **Activation Functions**

Activation functions are attached to each neuron and are mathematical equations that determine whether a neuron should be activated or not based on whether the neuron’s input is relevant for the model’s prediction or not. The purpose of the activation function is to introduce the nonlinearity in the data.

**Various Types of Activation Functions are :**

* Sigmoid Activation Function
* TanH / Hyperbolic Tangent Activation Function
* Rectified Linear Unit Function (ReLU)
* Leaky ReLU
* Softmax

## **What Is a Convolutional Neural Network?**

A convolutional neural network is one adapted for analysing and identifying visual data such as digital images or photographs.

## **What Is a Recurrent Neural Network?**

A recurrent neural network is one adapted for analysing time series data, event history, or temporal ordering.

## **What Is a Deep Neural Network?**

Also known as a deep learning network, is one that involves two or more processing layers.

## **Types of Artificial Neural Network:**

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. For example, segmentation or classification.

### Feedback ANN:

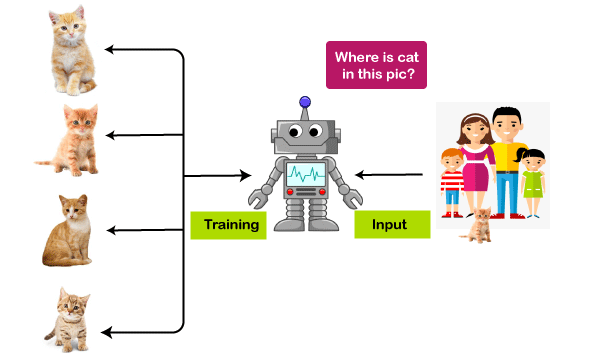
The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

### Feed-Forward ANN:

A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behaviour of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

### Importance of Neural Network:

* **Without Neural Network:**

Let's have a look at the example given below. Here we have a machine, such that we have trained it with four types of cats, as you can see in the image below. And once we are done with the training, we will provide a random image to that particular machine that has a cat. Since this cat is not similar to the cats through which we have trained our system, so without the neural network, our machine would not identify the cat in the picture. Basically, the machine will get confused in figuring out where the cat is.  
  


* **With Neural Network:**

However, when we talk about the case with a neural network, even if we have not trained our machine with that particular cat. But still, it can identify certain features of a cat that we have trained on, and it can match those features with the cat that is there in that particular image and can also identify the cat.

## **Advantages of Artificial Neural Network**

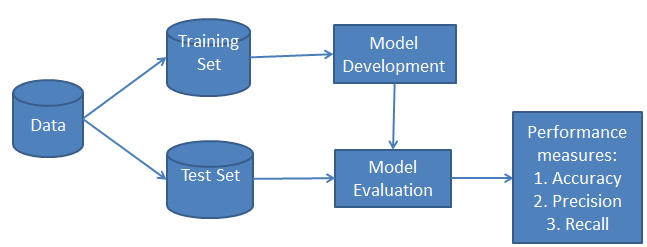
* **Parallel processing capability**
* **Storing data on the entire network**
* **Capability to work with incomplete knowledge**
* **Having a memory distribution**
* **Having fault tolerance**

## **Disadvantages of Artificial Neural Network**

* **Assurance of proper network structure**
* **Unrecognized behaviour of the network**
* **Hardware dependence**
* **Difficulty of showing the issue to the network**
* **The duration of the network is unknown**

# **4.8. TRAINING, TESTING AND EVALUATING MACHINE LEARNING MODELS**

# **Introduction to Model Evaluation**

In Machine Learning, our goal is to achieve a machine learning model that generalizes well on new unseen data or unknown data. There is always a problem of overfitting while training your machine learning model which is a central obstacle to take care of. While some of you may have a small dataset and training neural networks on small datasets doesn’t generalize well and began to overfit. It’s very important to measure the generalization power of your model so that it can perform well for what it was trained for avoiding overfitting case.

# **Training, Validation and Test Sets**

While evaluating a model we always split our data into three sets:

* Training
* Validation
* Test set.

First train the model on the training data and evaluate the validation data and once the model is ready, test it one final time on the test data.

**Training set** – refers to a subset of a dataset used to build predictive models. It includes a set of input examples that will be used to train a model by adjusting the parameters of the set.

**Validation set** – is a subset of a dataset whose purpose is to assess the performance of the model built, during the training phase. It periodically evaluates a model and allows for fine-tuning of the parameters of the model.

**Test set** – this is also known as unseen data. It is the final evaluation that a model undergoes after the training phase. A test set is a subset of a dataset used to assess the possible future performance of a model. For example, if a model fits the training better than the test set, overfitting is likely present.

**Overfitting**– refers to when a model contains more parameters than can be accounted for by the dataset. Noisy data contributes to overfitting. The generalization of these models is unreliable since the model learns more than it is meant to from the dataset.

**Common split percentages include:**

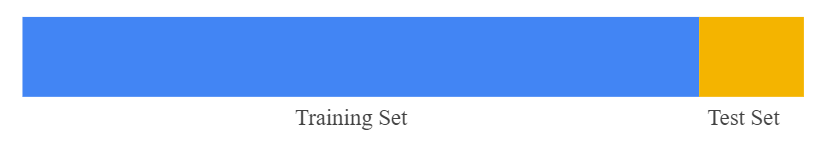
* Train: 80%, Test: 20%
* Train: 67%, Test: 33%
* Train: 50%, Test: 50%

## **Why not only training and test set, Why Validation?**

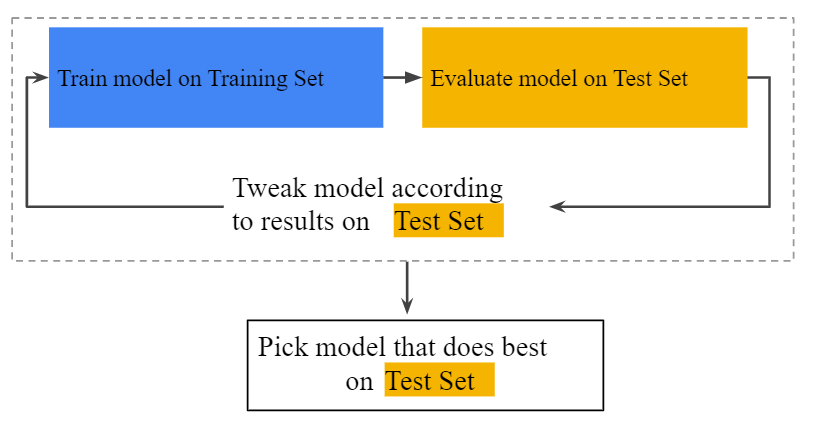
We can train on the training set and evaluate it on the test set and model will be ready, then why validation set?

## **Workflow without a validation set**

We divide our dataset into two parts :



Then we follow this process :



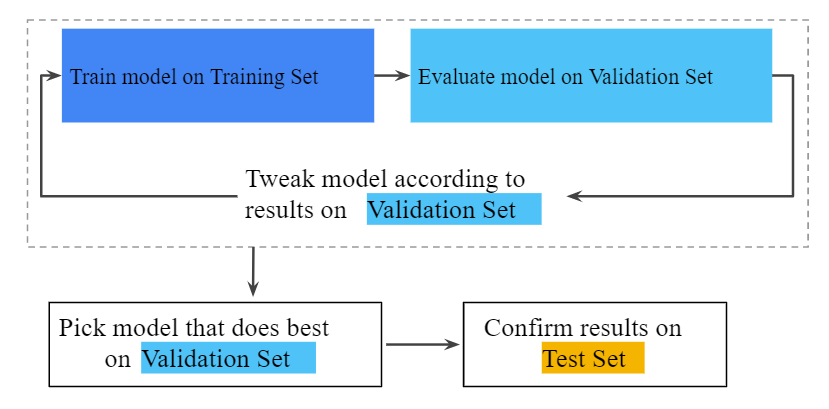
We train a model on a training set and then evaluate it on a test set and according to the result we tune our model parameters and then after the different iterative process on tuning model, we pick a model that does best on the test set.

**Why the Validation set?**

To reduce overfitting, a new set called validation set is used. In this approach, we divide our dataset into three sets: training, testing and validation sets.



And then we follow this process :



We train our model on the training set and this time we evaluate our model on the validation set and tune the parameters of the model according to the validation loss and accuracy and we repeat this process until we get the model that best fit on the validation set. And after choosing the best model we test or confirm our results on testing set to get the correct accuracy or how well our model is generalised.

### Model evaluation techniques

The techniques to evaluate the performance of a model can be divided into two parts: cross-validation and holdout. Both these techniques make use of a test set to assess model performance.

#### **Cross validation :**

Cross-validation involves the use of a training dataset and an independent dataset. These two sets result from partitioning the original dataset. The sets are used to evaluate an algorithm.

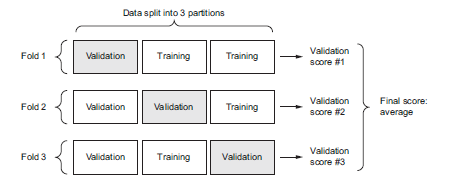
First, we split the dataset into groups of instances equal in size. These groups are called folds. The model to be evaluated is trained on all the folds except one. After training, we test the model on the fold that was excluded. This process is then repeated over and over again, depending on the number of folds.

If there are six folds, we will repeat the process six times. The reason for the repetition is that each fold gets to be excluded and act as the test set. Last, we measure the average performance across all folds to get an estimation of how effective the algorithm is on a problem.

A popular cross-validation technique is the k-fold cross-validation. It uses the same steps described above. The k, (is a user-specified number), stands for the number of folds. The value of k may vary based on the size of the dataset but as an example, let us use a scenario of 3-fold cross-validation.

The model will be trained and tested three times. Let’s say the first-round trains on folds 1 and 2 . The testing will be on fold 3. For the second round, it may train on folds 1 and 3 and test on fold 2. For the last round, it may train on folds 2 and 3 and test on fold 1.

The interchange between training and test data makes this method very effective. However, compared to the holdout technique, cross-validation takes more time to run and uses more computational resources.



#### **Holdout**

It’s important to get an unbiased estimate of model performance. This is exactly what the holdout technique offers. To get this unbiased estimate, we test a model on data different from the data we trained it on. This technique divides a dataset into three subsets: training, validation, and test sets.

As mentioned earlier, that the training set helps the model make predictions and that the test set assesses the performance of the model. The validation set also helps to assess the performance of the model by providing an environment to fine-tune the parameters of the model. From this, we select the best performing model.

The holdout method is ideal when dealing with a very large dataset, it prevents model overfitting, and incurs lower computational costs.

When a function fits too tightly to a set of data points, an error known as overfitting occurs. As a result, a model performs poorly on unseen data. To detect overfitting, we could first split our dataset into training and test sets. We then monitor the performance of the model on both training data and test data.

If our model offers superior performance on the training set when compared to the test set, there’s a good chance overfitting is present. For instance, a model might offer 90% accuracy on the training set yet give 50% on the test set.

### Model evaluation metrics

#### **a. Metrics for classification problems**

Predictions for classification problems yield four types of outcomes: true positives, true negatives, false positives, and false negatives.

##### **Classification accuracy**

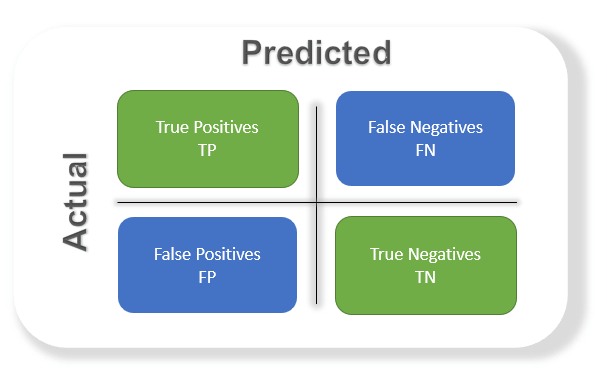
The most common evaluation metric for classification problems is accuracy. It’s taken as the number of correct predictions against the total number of predictions made (or input samples).

Classification accuracy works best if the samples belonging to each class are equal in number. Consider a scenario with 97% samples from class X and 3% from class Y in a training set. A model can very easily achieve 97% training accuracy by predicting each training sample in class X.

Testing the same model on a test set with 55% samples of X and 45% samples of Y, the test accuracy is reduced to 55%. This is why classification accuracy is not a clear indicator of performance. It provides a false sense of attaining high levels of accuracy.

##### **Confusion matrix**

The confusion matrix forms the basis for the other types of classification metrics. It’s a matrix that fully describes the performance of the model. A confusion matrix gives an in-depth breakdown of the correct and incorrect classifications of each class.



Confusion Matrix Terms:

* **True positives** are when you predict an observation belongs to a class and it actually does belong to that class.
* **True negatives** are when you predict an observation does not belong to a class and it actually does not belong to that class.
* **False positives** occur when you predict an observation belongs to a class when in reality it does not.
* **False negatives** occur when you predict an observation does not belong to a class when in fact it does.

The confusion matrix explained above is an example for the case of binary classification.

  From this it’s important to amplify true positives and true negatives. False positives and false negatives represent misclassification, that could be costly in real-world applications. Consider instances of misdiagnosis in a medical deployment of a model.

A model may wrongly predict that a healthy person has cancer. It may also classify someone who actually has cancer as cancer-free. Both these outcomes would have unpleasant consequences in terms of the well being of the patients after being diagnosed (or finding out about the misdiagnosis), treatment plans as well as expenses. Therefore it’s important to minimize false negatives and false positives.

The green shapes in the image represent when the model makes the correct prediction. The blue ones represent scenarios where the model made the wrong predictions. The rows of the matrix represent the actual classes while the columns represent predicted classes.

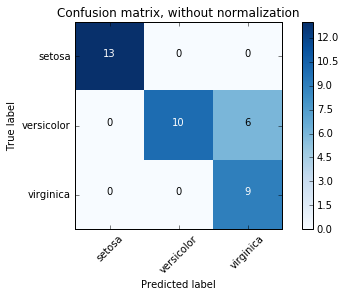
We can calculate accuracy from the confusion matrix. The accuracy is given by taking the average of the values in the “true” diagonal.

**Accuracy = (True Positive + True Negative) / Total Sample**

That translates to:

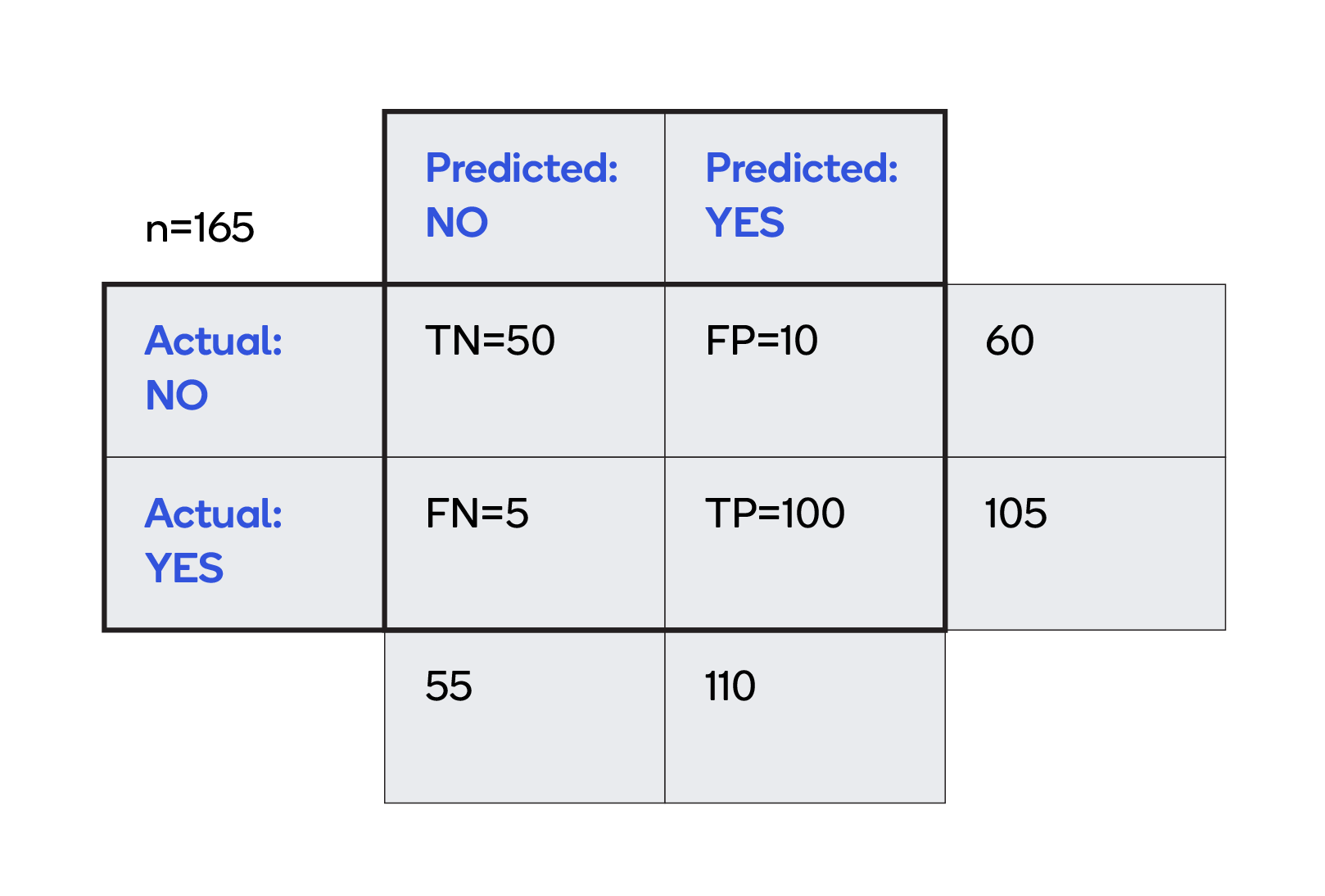
**Accuracy = Total Number of Correct Predictions / Total Number of Observations**

This can also extended to plot multi-class classification predictions like following example of classifying observations from the Iris flower dataset.



#### **Accuracy**

It is also useful to calculate the accuracy based on classifier prediction and actual value.



Accuracy is a measure of how often, over all observations, the classifier is correct.

Accuracy is : ***(TP+TN)/total = (100+50)/(165) = 0.91.***

* 1. **Precision**

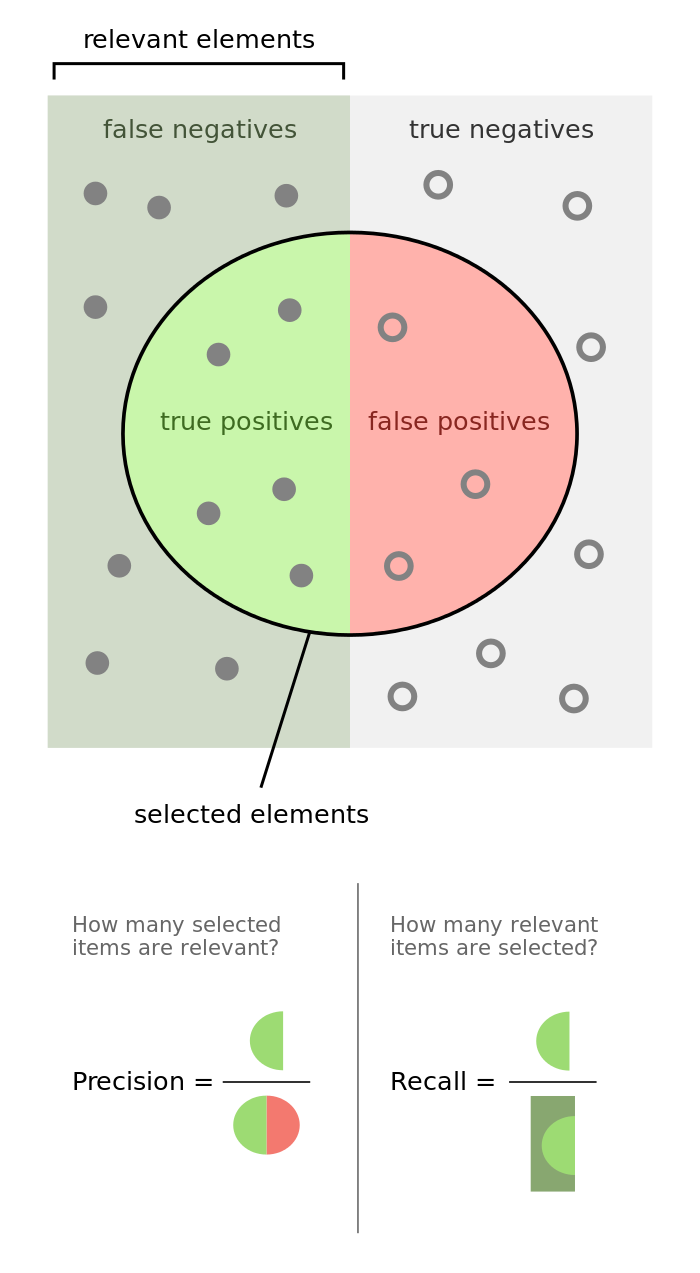
Precision refers to the number of true positives divided by the total positive results predicted by a classifier. That is, precision aims to understand what fraction of all positive predictions were actually correct.

**Precision = True Positives / (True Positives + False Positives)**

* 1. **Recall**

On the other hand, recall is the number of true positives divided by all the samples that should have been predicted as positive. Recall has the goal to perceive what fraction of actual positive predictions were identified accurately.

**Recall = True Positives / (True Positives + False Negatives)**



* 1. **F-score**

F-score is a metric that incorporates both the precision and recall of a test to determine the score. F-score is also known as F-measure or F1 score.

fscoreIn addition to robustness, the F-score shows us how precise a model is by letting us know how many correct classifications are made. The F-score ranges between 0 and 1. The higher the F-score, the greater the performance of the model.

#### **b. Metrics for regression problems**

Classification models deal with discrete data. The above metrics are ideal for classification tasks since they are concerned with whether a prediction is correct.

Regression models, on the other hand, deal with continuous data. Predictions are in a continuous range. This is the distinction between the metrics for classification and regression problems.

##### **Mean absolute error**

The mean absolute error represents the average of the absolute difference between the original and predicted values.

Mean absolute error provides the estimate of how far off the actual output the predictions were. However, since it’s an absolute value, it does not indicate the direction of the error.

maeMean absolute error is given by:

##### **Mean squared error**

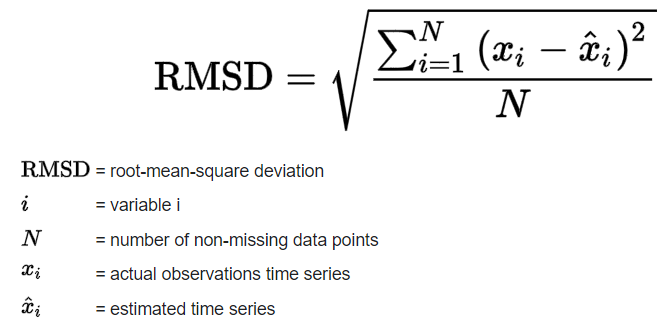
The mean squared error is quite similar to the mean absolute error. However, mean squared error uses the average of the square of the difference between original and predicted values. Since this involves the squaring of the errors, larger errors are very notable.

mseMean squared error is given by:

##### **Root mean squared error**

The root mean squared error (RMSE), computes the idealness of fit by calculating the square root of the average of squared differences between the predicted and actual values. It’s a measure of the average error magnitude.

The root mean squared error is a form of normalized distance between the vectors of the observed and predicted values.



**QUESTION BANK**

|  |  |  |  |
| --- | --- | --- | --- |
| **Part-A** | | | |
| **Q.No** | **Questions** | **Competence** | **BT Level** |
|  | Define Machine Learning. | Remember | BTL 1 |
|  | List the types of Machine Learning? Explain with example | Understand | BTL 2 |
|  | List the types of Reinforcement Learning. | Understand | BTL 2 |
|  | Define curse of Dimensionality. | Remember | BTL 1 |
|  | Define Entropy and Information Gain in Decision Tree algorithm | Remember | BTL 1 |
|  | Differentiate between Regression and Classification Algorithm. | Understand | BTL 2 |
|  | Illustrate Outlier, Overfitting and Underfitting? | Understand | BTL 2 |
|  | List the approaches of dimensionality reduction? | Understand | BTL 2 |
|  | Define PCA. | Remember | BTL 1 |
|  | Interpret Perceptron and Multi-Layer Perceptron? | Understand | BTL 2 |
|  | List the types of Activation functions? | Understand | BTL 2 |
|  | List the common splitting percentages of training and testing? | Understand | BTL 2 |
|  | Define feature selection | Understand | BTL 2 |
|  | Mention the two components of Bayesian Network. | Understand | BTL 2 |
|  | Define Neural Network and mention its components | Understand | BTL 2 |
|  | Differentiate between Precision and Recall. | Analysis | BTL 4 |
|  | List the different methods in feature selection? | Understand | BTL 2 |
|  | How F-score is calculated? | Analysis | BTL 4 |
|  | List the types of ANN | Apply | BTL 3 |
|  | Define Confusion Matrix? | Understand | BTL 2 |
| **PART B** | | | |
| **Q.No** | **Questions** | **Competence** | **BT Level** |
|  | Explain the three types of Machine Learning. | Analysis | BTL 4 |
|  | Compute the principal component of following data : (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8). | Apply | BTL 3 |
|  | Explain Linear Regression with example. | Analysis | BTL 4 |
|  | Explain the following terms in Bayesian Network:   * 1. DAG   2. Components of Bayesian network   3. Joint Probability Distribution | Analysis | BTL 4 |
|  | Discuss about Decision tree? Explain its structure and steps involved in ID3 algorithm. | Analysis | BTL 4 |
|  | Explain the working of neural network with its architecture | Analysis | BTL 4 |
|  | Explain the need of Validation set with its working flow? | Analysis | BTL 4 |
|  | 1. Explain the following Evaluation Metrics:    1. Classification Accuracy    2. Confusion Matrix    3. Mean absolute error    4. Mean squared error | Analysis | BTL 4 |
|  | Explain the methods to test and validate the machine learning model? | Analysis | BTL 4 |
|  | Explain about the Bayesian network with its building models? | Analysis | BTL 4 |