UNIT-III

Find-S: Finding a maximally specific hypothesis:

Introduction:

The find-S algorithm is a basic concept learning algorithm in machine learning. The find-S algorithm finds the most specific hypothesis that fits all the positive examples. We have to note here that the algorithm considers only those positive training example. The find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive training data. Hence, the Find-S algorithm moves from the most specific hypothesis to the most general hypothesis.

Important Representation :

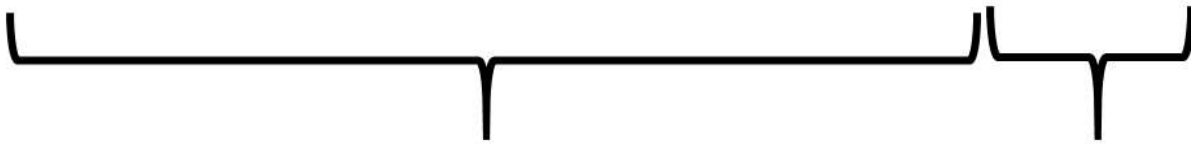
1. **?** indicates that any value is acceptable for the attribute.
2. specify a single required value (e.g., Cold) for the attribute.
3. **Φ** indicates that no value is acceptable.
4. The most **general hypothesis** is represented by: **{?, ?, ?, ?, ?, ?}**
5. The most **specific hypothesis** is represented by: **{ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}**

Steps Involved In Find-S:

1. Start with the most specific hypothesis. **h = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}**
2. Take the next example and if it is negative, then no changes occur to the hypothesis.
3. If the example is positive and we find that our initial hypothesis is too specific then we update our current hypothesis to a general condition.
4. Keep repeating the above steps till all the training examples are complete.
5. After we have completed all the training examples we will have the final hypothesis when can use to classify the new examples.

Consider the following data set having the data about which particular seeds are poisonous.

EXAMPLE	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES



ATTRIBUTES ON WHICH THE CONCEPT DEPENDS ON CONCEPT

First, we consider the hypothesis to be a more specific hypothesis. Hence, our hypothesis would be :

h = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}

Consider example 1 :

The data in example 1 is { GREEN, HARD, NO, WRINKLED }. We see that our initial hypothesis is more specific and we have to generalize it for this example. Hence, the hypothesis becomes :

h = { GREEN, HARD, NO, WRINKLED }

Consider example 2 :
Here we see that this example has a negative outcome. Hence we

neglect this example and our hypothesis remains the same.

h = { GREEN, HARD, NO, WRINKLED }

Consider example 3 :
Here we see that this example has a negative outcome. Hence we neglect this example and our hypothesis remains the same.
h = { GREEN, HARD, NO, WRINKLED }

Consider example 4 :
The data present in example 4 is { ORANGE, HARD, NO, WRINKLED }. We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (” ? ”). After doing the process the hypothesis becomes :
h = { ?, HARD, NO, WRINKLED }

Consider example 5 :
The data present in example 5 is { GREEN, SOFT, YES, SMOOTH }. We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (” ? ”). After doing the process the hypothesis becomes :
h = { ?, ?, ?, ? }

Since we have reached a point where all the attributes in our hypothesis have the general condition, example 6 and example 7 would result in the same hypothesizes with all general attributes.
h = { ?, ?, ?, ? }

Hence, for the given data the final hypothesis would be :

Final Hyposthesis: h = { ?, ?, ?, ? }

Algorithm :

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x

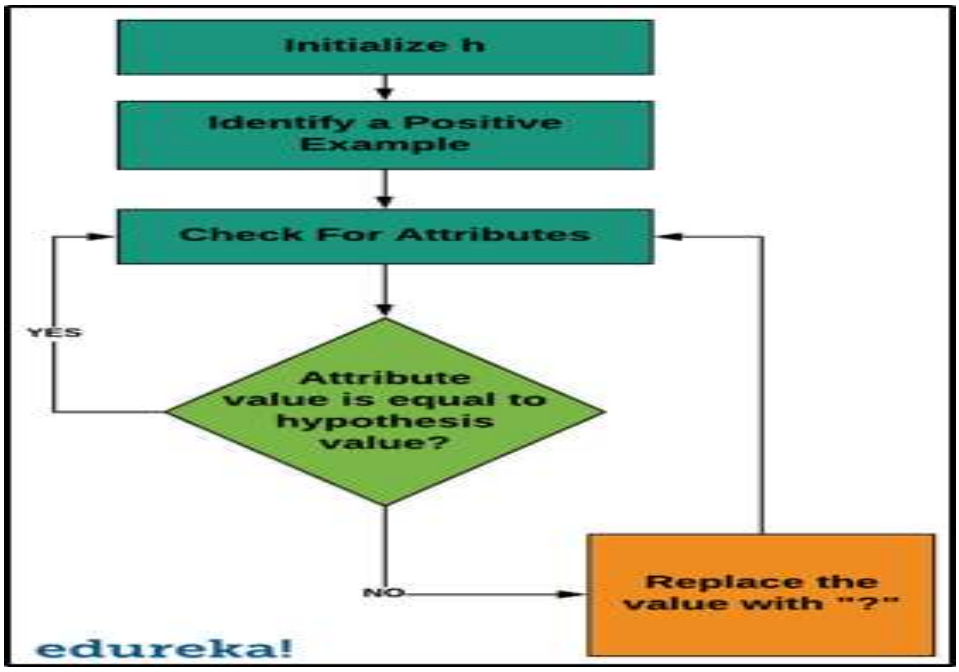
For each attribute constraint a, in h

If the constraint a, is satisfied by x

Then do nothing

Else replace a, in h by the next more general constraint that is satisfied by x

3. Output hypothesis h

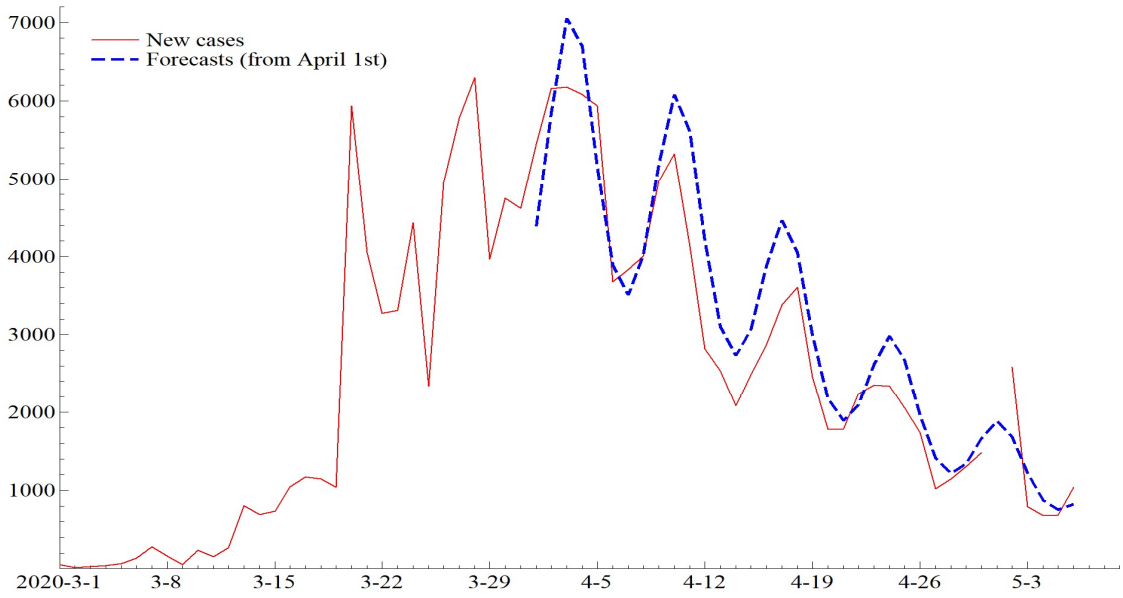


Analysis of Time Series:

We know that planning about future is very necessary for the every business firm, every govt. institute, every individual and every country. Time Series data can be found in many applications economics, social sciences, finance, epidemiology, and the physical sciences.

It is a statistical technique deals with time series data and trend analysis. Time Series data means data in a particular time periods or intervals. Time series data is data that is collected at different points in time. Data points in time series are collected at adjacent time periods there is potential for correlation between observations.

Example : Global air pollution from the Countries, unemployment rates in India etc, Disease prediction in countries.



A time series is set of observation taken at specified times, usually at equal intervals Mathematically a time series is defined by the values Y1, Y2, ….Yn of variable Y at times t1,t2,…,tn

$$Y=f(t)$$

Objectives of Time series analysis

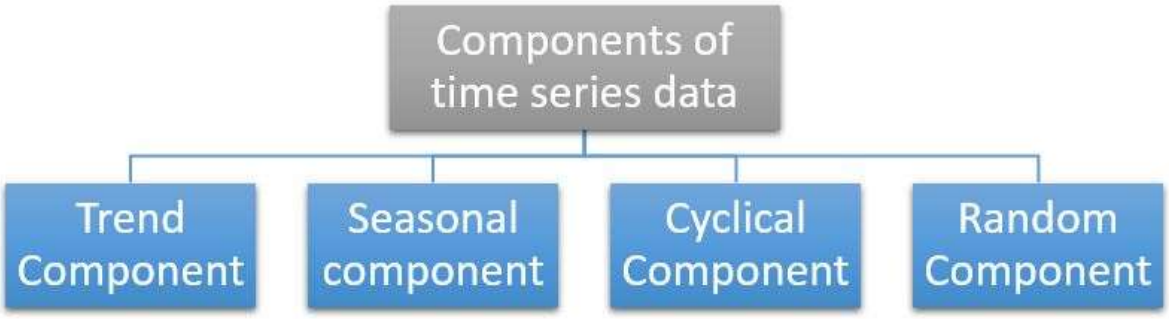
- To describe the important features of the time series pattern
- To explain how the past affects the future or how two time series can "interact".
- To forecast future values of the series.
- To possibly serve as a control standard for a variable that measures the quality of product in some manufacturing situations.

Importance of Time Series Analysis:

- A popular tool for Business forecasting
- Basis for understanding past behaviour

- Can forecast future activities/ Planning future operations
- Evaluate current accomplishments/ evaluation of performance

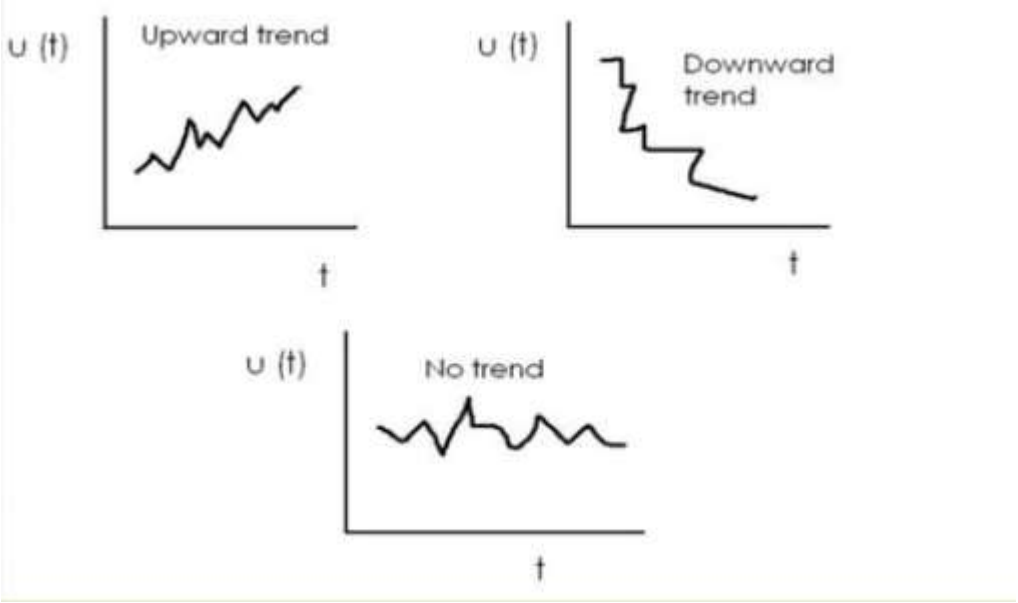
Components in Time Series:



Trend Component:

Trend is a pattern in data that shows the movement of a series to relatively higher or lower values over a long period of time. In other words, a trend is observed when there is an increasing or decreasing slope in the time series. Trend usually happens for some time and then disappears, it does not repeat. For example, some new song comes, it goes trending for a while, and then disappears. There is fairly any chance that it would be trending again.
A trend could be :

- **Uptrend:** Time Series Analysis shows a general pattern that is upward then it is Uptrend.
- **Downtrend:** Time Series Analysis shows a pattern that is downward then it is Downtrend.
- **Horizontal or Stationary trend:** If no pattern observed then it is called a Horizontal or stationary trend.



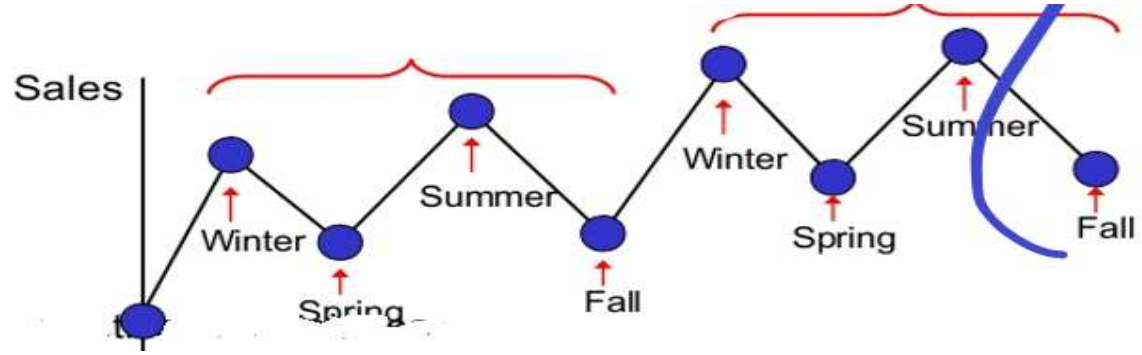
Seasonal Component:

The seasonal component is that part of the variations in a time series representing intra-year fluctuations that are more or less stable year after year with respect to timing, direction and magnitude.

The seasonal component is also referred to as the seasonality of a time series.

The seasonal component reflect “normal” variations that recur every year to the same extent, e.g. weather fluctuations that are representative of the season, length of months, Christmas effect.

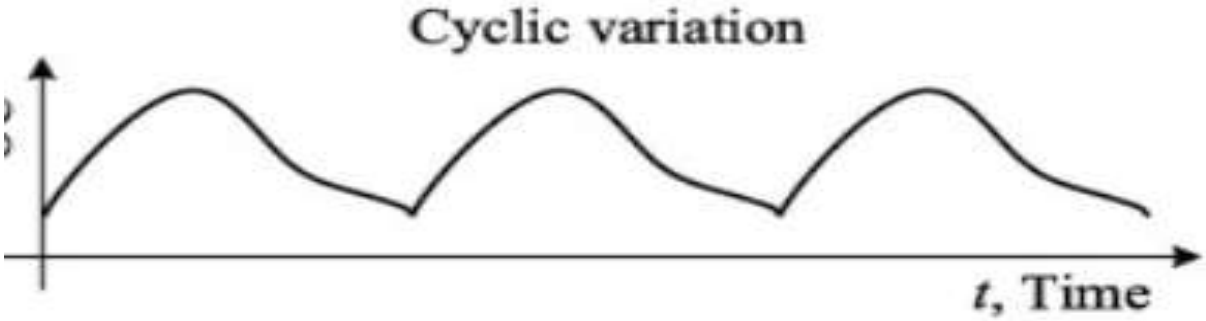
The seasonal component may also include calendar related systematic effects that are not regular in their annual timing or are caused by variations in the calendar from year to year.



Cyclic Variations

The cyclical component of a time series refers to (regular or periodic) fluctuations around the trend, excluding the irregular component, revealing a succession of phases of expansion and contraction.

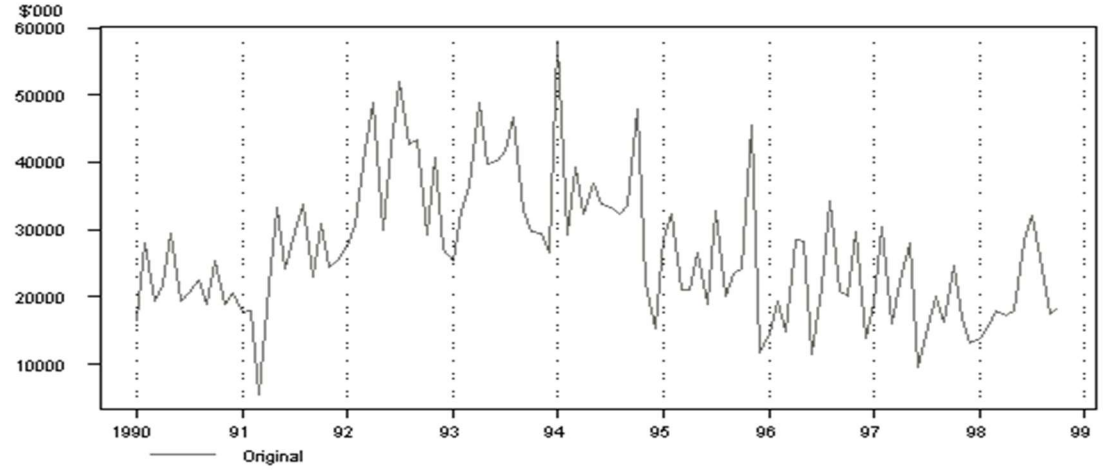
The cyclical component can be viewed as those fluctuations in a time series which are longer than a given threshold, e.g. 1½ years, but shorter than those attributed to the trend.



IRREGULAR Component:

The irregular component (sometimes also known as the residual) is what remains after the

seasonal and trend components of a time series have been estimated and removed. It results from short term fluctuations in the series which are neither systematic nor predictable. In a highly irregular series, these fluctuations can dominate movements, which will mask the trend and seasonality.



Additive Decomposition

In some time series, the amplitude of both the seasonal and irregular variations do not change as the level of the trend rises or falls. In such cases, an additive model is appropriate.

In the additive model, the observed time series (O_t) is considered to be the sum of three independent components: the seasonal S_t, the trend T_t and the irregular I_t.

$$\text{Observed series} = \text{Trend} + \text{Seasonal} + \text{Irregular}$$

That is

$$O_t = T_t + S_t + I_t$$

Each of the three components has the same units as the original series. The seasonally adjusted series is obtained by estimating and removing the seasonal effects from the original time series. The estimated seasonal component is denoted by $\hat{S}_t$ The seasonally adjusted estimates can be expressed by:

$$\begin{aligned}\text{Seasonally adjusted series} &= \text{Observed series} - \text{Seasonal} \\ &= \text{Trend} + \text{Irregular}\end{aligned}$$

In symbols,

$$\begin{aligned}SA_t &= O_t - \hat{S}_t \\ &= T_t + I_t\end{aligned}$$

Multiplicative Decomposition

In many time series, the amplitude of both the seasonal and irregular variations increase as the level of the trend rises. In this situation, a multiplicative model is usually appropriate.

In the multiplicative model, the original time series is expressed as the product of trend, seasonal and irregular components.

$$\text{Observed series} = \text{Trend} \times \text{Seasonal} \times \text{Irregular}$$

or

$$O_t = T_t \times S_t \times I_t$$

The seasonally adjusted data then becomes:

$$\begin{aligned}\text{Seasonally Adjusted series} &= \text{Observed} \div \text{Seasonal} \\ &= \text{Trend} \times \text{Irregular}\end{aligned}$$

or

$$\begin{aligned}SA_t &= O_t / \hat{S}_t \\ &= T_t \times I_t\end{aligned}$$

Linear time Series Analysis:

Linear time series analysis provides a natural framework to study the dynamic structure of such a series. The theories of linear time series discussed in the chapter include stationarity, dynamic dependence, autocorrelation function, modeling, and forecasting.

The econometric models introduced in the chapter include: (a) simple autoregressive (AR) models, (b) simple moving-average (MA) models, (c) mixed autoregressive moving-average (ARMA) models

a) Single Autoregressive:

- Autoregression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step.
- A regression model, such as linear regression, models an output value based on a linear combination of input values.

  yhat = b0 + b1*X1

This technique can be used on time series where input variables are taken as observations at previous time steps, called lag variables.

$$X(t+1) = b0 + b1*X(t-1) + b2*X(t-2)$$

b) Simple moving-average (MA) models:

The forecast for the value of Y at time t+1 that is made at time t equals the simple average of the most recent m observations:

$$\hat{Y}_{t+1} = \frac{Y_t + Y_{t-1} + \ldots + Y_{t-m+1}}{m}$$

This average is *centered* at period t-(m+1)/2, which implies that the estimate of the local mean will tend to lag behind the true value of the local mean by about (m+1)/2 periods.

Thus, we say **the *average age* of the data in the simple moving average is (m+1)/2** relative to the period for which the forecast is computed: this is the amount of time by which forecasts will tend to *lag behind turning points* in the data.

c) Auto-Regressive Integrated Moving Average:

An ARIMA model can be understood by outlining each of its components as follows:

Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.

Integrated *(I)*: represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values.

Moving average (MA): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

NonLinear Dynamics:

Nonlinear dynamics is the branch of physics that studies systems governed by equations more complex than the linear, aX+b form. Nonlinear systems, such as the weather or neurons, often appear chaotic, unpredictable or counterintuitive, and yet their behaviour is not random.

A nonlinear system is a system in which the change of the output is not proportional to the change of the input. Nonlinear dynamical systems, describing changes in variables over time, may appear chaotic, unpredictable, or counterintuitive, contrasting with much simpler linear systems.

The behavior of a nonlinear system is described in mathematics by a nonlinear system of equations, which is a set of simultaneous equations in which the unknowns appear as variables of a polynomial of degree higher than one or in the argument of a function which is not a polynomial of degree one.

Nonlinear ordinary differential equations occur in numerous forms depending on the nature of the problem.

These problems could be either Hamiltonian or dissipative.

The solution properties of the underlying nonlinear differential equations like integrability, non-integrability, chaos, localization, patterns, etc. depend upon the order of the highest derivative, number of dependent variables, degree of nonlinearity, whether homogeneous or inhomogeneous and the numerical values of the controlling parameters as well as on the initial conditions.

Rule Based Induction:

Rule induction is one of the most important techniques of machine learning. Since regularities hidden in data are frequently expressed in terms of rules, rule induction is one of the fundamental tools of data mining at the same time.

Usually rules are expressions of the form

if (attribute − 1, value − 1)

and (attribute − 2, value − 2)

and ⋯

and (attribute − n, value − n)

then (decision, value)

Some rule induction systems induce more complex rules, in which values of attributes may be expressed by negation of some values or by a value subset of the attribute domain

Table 1.1. An Example of a Dataset.

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no

Table 1.2. An Example of an Erroneous Dataset

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	42.5	no	no

- A case x is covered by a rule r if and only if every condition of r is satisfied by the corresponding attribute value for x
- A concept C is completely covered by a rule set R if and only if for every case x from C there exists a rule r from R such that r covers x.
- A rule r is consistent if and only if for every case x covered by r, x is a member of the concept C indicated by r. A rule set R is consistent if and only if every rule from R is consistent with the data set.

Rule Induction Rule Induction Using Sequential Covering Algorithm:

Sequential Covering Algorithm can be used to extract IF-THEN rules form the training data.

As per the general strategy the rules are learned one at a time. For each time rules are learned, a tuple covered by the rule is removed and the process continues for the rest of the tuples.

The sequential learning Algorithm where rules are learned for one class at a time. When learning a rule from a class Ci, we want the rule to cover all the tuples from class C only and no tuple form any other class.

Algorithm: Sequential Covering

Input:

- D, a data set class-labeled tuples,
- Att_vals, the set of all attributes and their possible values.

Output: A Set of IF-THEN rules.

Method:

- Rule_set={ }; // initial set of rules learned is empty
- for each class c do
- repeat
- Rule = Learn_One_Rule(D, Att_valls, c);

- remove tuples covered by Rule form D;
- until termination condition;
- Rule_set=Rule_set+Rule; // add a new rule to rule-set
- end for
- return Rule_Set;

Table 1.1. An Example of a Dataset.

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no

Case 1: (Headache, yes) → (Flu, yes)

From cases 2,4,5 is (Headache, yes) → (Flu, yes) accepted?

Therefore it is not consistent..

If r is (Headache, yes)&(Weakness, yes) → (Flu, yes) &

(Temperature, high)&(Headache, yes) → (Flu, yes) is this consistent ?

case 5 is not covered by any rule.

Task of rule induction is to induce a rule set R that is consistent and complete.

(Headache, yes) → (Flu, yes),

(Temperature, high)&(Weakness, yes) → (Flu, yes),

(Temperature, normal)&(Headache, no) → (Flu, no),

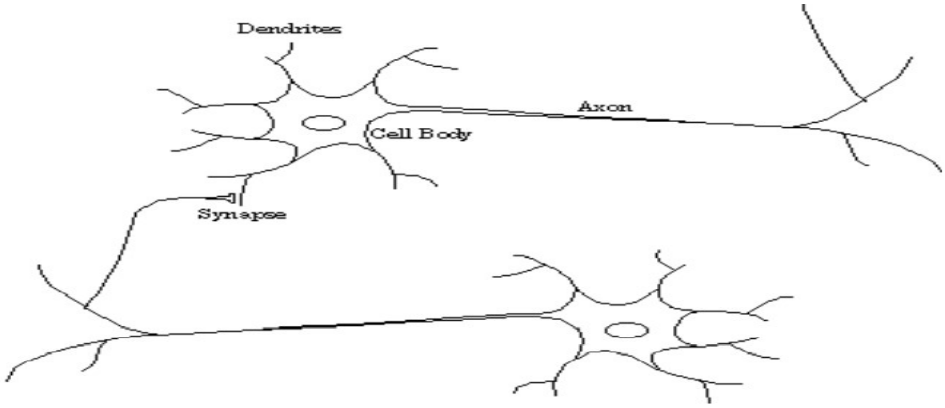
(Headache, no)&(Weakness, no) → (Flu, no). is deterministic.

Human Brain:

Humans perform complex tasks like vision, motor control, or language understanding very well. One way to build intelligent machines is to try to imitate the human brain. The brain is a highly complex, non-linear, and parallel computer, composed of some 10^{11} neurons that are densely connected ($\sim 10^4$ connection per neuron).

A neuron is much slower (10^{-3}sec) compared to a silicon logic gate (10^{-9}sec), however the massive interconnection between neurons make up for the comparably slow rate.

Complex perceptual decisions are arrived at quickly (within a few hundred milliseconds)



The interneuron connections are mediated by electrochemical junctions called synapses, which are located on branches of the cell referred to as dendrites. Each neuron typically receives many thousands of connections from other neurons and is therefore constantly receiving a multitude of incoming signals, which eventually reach the cell body.

If the resulting signal exceeds some threshold then the neuron will "fire" or generate a voltage impulse in response. This is then transmitted to other neurons via a branching fibre known as the axon.

Neural Network:

Computational models inspired by the human brain:

– Massively parallel, distributed system, made up of simple processing units (neurons)

– Synaptic connection strengths among neurons are used to store the acquired knowledge.

– Knowledge is acquired by the network from its environment through a learning process

Properties of Neural Network

Learning from examples – labeled or unlabeled

Adaptivity – changing the connection strengths to learn things

Non-linearity – the non-linear activation functions are essential

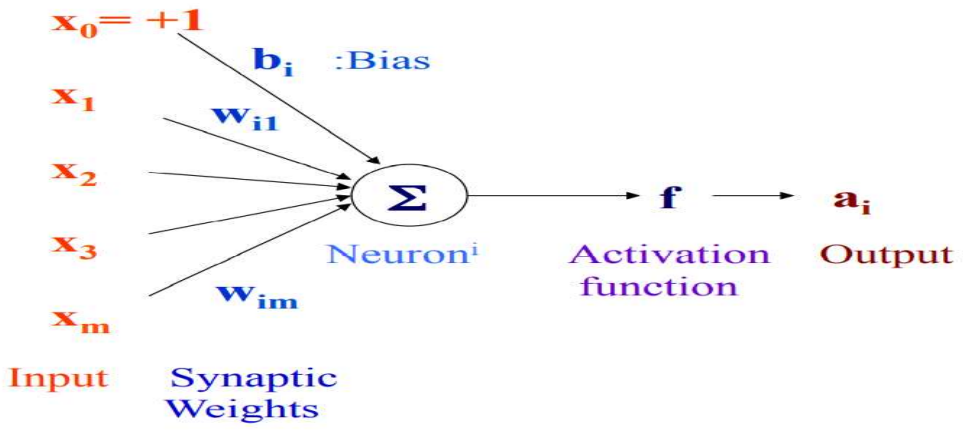
Fault tolerance – if one of the neurons or connections is damaged, the whole network still works quite well

Thus, they might be better alternatives than classical solutions for problems characterised by:

– high dimensionality, noisy, imprecise or imperfect data; and

– a lack of a clearly stated mathematical solution or algorithm Neuron Mode.

Artificial Neuron



An artificial neuron is a connection point in an artificial neural network. Artificial neural networks, like the human body's biological neural network, have a layered architecture and each network node (connection point) has the capability to process input and forward output to other nodes in the network. In both artificial and biological architectures, the nodes are called neurons and the connections are characterized by synaptic weights, which represent the significance of the connection. As new data is received and processed, the synaptic weights change and this is how learning occurs.

Artificial neurons are modeled after the hierarchical arrangement of neurons in biological sensory systems. In the visual system, for example, light input passes through neurons in successive layers of the retina before being passed to neurons in the thalamus of the brain and then on to neurons in the brain's visual cortex. As the neurons pass signals through an increasing number of layers, the brain progressively extracts more information until it is confident it can identify what the person is seeing. In artificial intelligence, this fine tuning process is known as deep learning.

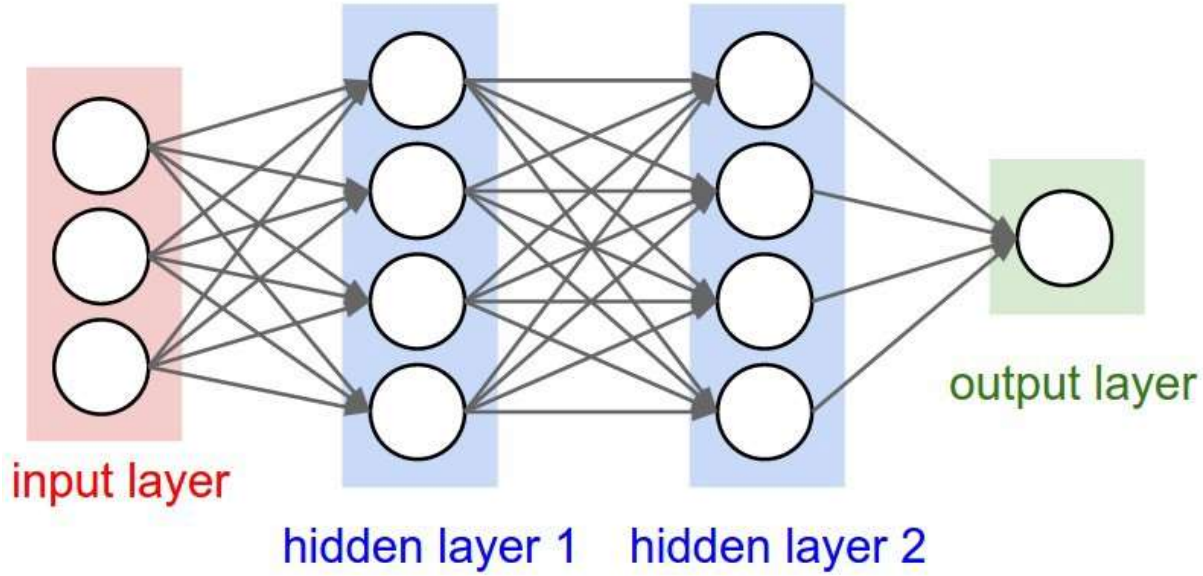
In both artificial and biological networks, when neurons process the input they receive, they decide whether the output should be passed on to the next layer as input. The decision of whether or not to send information on is called bias and it’s determined by an activation function built into the system. For example, an artificial neuron may only pass an output signal on to the next layer if its inputs (which are actually voltages) sum to a value above some particular threshold value. Because activation functions can either be linear or non-linear, neurons will often have a wide range of convergence and divergence. Divergence is the ability for one neuron to communicate with many other neurons in the network and convergence is the ability for one neuron to receive input from many other neurons in the network.

Neural Network:

Neural Networks can approximate as well as learn and represent any function given a large enough layer and desired error margin. The way neural network learns the true function is by building complex representations on top of simple ones. On each hidden layer, the neural network learns new feature space by first compute the affine (linear) transformations of the given inputs and then apply non-linear function which in turn will be the input of the next layer. This process will continue until we reach the output layer. Therefore, we can define neural network as information flows from inputs through hidden layers towards the output. For a 3-layers neural network, the learned function would be: $f(x) = f\_3(f\_2(f\_1(x)))$ where:

- $f\_1(x)$: Function learned on first hidden layer
- $f\_2(x)$: Function learned on second hidden layer
- $f\_3(x)$: Function learned on output layer

Therefore, on each layer we learn different representation that gets more complicated with later hidden layers.Below is an example of a 3-layers neural network (we don't count input layer):



For example, computers can't understand images directly and don't know what to do with pixels data. However, a neural network can build a simple representation of the image in the early hidden layers that identifies edges. Given the first hidden layer output, it can learn corners and contours. Given the second hidden layer, it can learn parts such as nose. Finally, it can learn the object identity.

Forward Propagation:

Forward Propagation

The input *X* provides the initial information that then propagates to the hidden units at each layer and finally produce the output y^. The architecture of the network entails determining its depth, width, and activation functions used on each layer. **Depth** is the number of hidden layers. **Width** is the number of units (nodes) on each hidden layer since we don't control neither input layer nor output layer dimensions. There are quite a few set of activation functions such *Rectified Linear*

Unit, Sigmoid, Hyperbolic tangent, etc. Research has proven that deeper networks outperform networks with more hidden units. Therefore, it's always better and won't hurt to train a deeper network (with diminishing returns).

Lets first introduce some notations that will be used throughout the post:

- W^l: Weights matrix for the l^{th} layer
- b^l: Bias vector for the l^{th} layer
- Z^l: Linear (affine) transformations of given inputs for the l^{th} layer
- g^l: Activation function applied on the l^{th} layer
- A^l: Post-activation output for the l^{th} layer
- dW^l: Derivative of the cost function w.r.t W^l $(\frac{\partial J}{\partial W^l})$
- db^l: Derivative of the cost function w.r.t b^l $(\frac{\partial J}{\partial b^l}))$
- dZ^l: Derivative of the cost function w.r.t Z^l $(\frac{\partial J}{\partial Z^l})$
- dA^l: Derivative of the cost function w.r.t A^l $(\frac{\partial J}{\partial A^l})$
- n^l: Number of units (nodes) of the l^{th} layer
- m: Number of examples
- L: Number of layers in the network (not including the input layer)

Next, we'll write down the dimensions of a multi-layer neural network in the general form to help us in matrix multiplication because one of the major challenges in implementing a neural network is getting the dimensions right.

- W^l, dW^l: Number of units (nodes) in l^{th} layer x Number of units (nodes) in $l-1$ layer
- b^l, db^l: Number of units (nodes) in l^{th} layer x 1
- Z^l, dZ^l: Number of units (nodes) in l^{th} layer x number of examples
- A^l, dA^l: Number of units (nodes) in l^{th} layer x number of examples

The two equations we need to implement forward propagations are: These computations will take place on each layer.

Parameters Initialization

We'll first initialize the weight matrices and the bias vectors. It's important to note that we shouldn't initialize all the parameters to zero because doing so will lead the gradients to be equal and on each iteration the output would be the same and the learning algorithm won't learn anything. Therefore, it's important to randomly initialize the parameters to values between 0 and 1. It's also recommended to multiply the random values by small scalar such as 0.01 to make the activation units active and be on the regions where activation functions' derivatives are not close to zero.

Activation Functions

There is no definitive guide for which activation function works best on specific problems. It's a trial and error process where one should try different set of functions and see which one works best on the problem at hand. We'll cover 4 of the most commonly used activation functions:

- **Sigmoid function (σ)**: *g(z) = 1 / (1 + e^{-z})*. It's recommended to be used only on the output layer so that we can easily interpret the output as probabilities since it has restricted output between 0 and 1. One of the main disadvantages for using sigmoid function on hidden layers is that the gradient is very close to zero over a large portion of its domain which makes it slow and harder for the learning algorithm to learn.

- **Hyperbolic Tangent function**: $g(z) = (e^z - e^{-z}) / (e^z + e^{-z})$. It's superior to sigmoid function in which the mean of its output is very close to zero, which in other words center the output of the activation units around zero and make the range of values very small which means faster to learn. The disadvantage that it shares with sigmoid function is that the gradient is very small on good portion of the domain.
- **Rectified Linear Unit (ReLU)**: $g(z) = max\{0, z\}$. The models that are close to linear are easy to optimize. Since ReLU shares a lot of the properties of linear functions, it tends to work well on most of the problems. The only issue is that the derivative is not defined at $z = 0$, which we can overcome by assigning the derivative to 0 at $z = 0$. However, this means that for $z \leq 0$ the gradient is zero and again can't learn.
- **Leaky Rectified Linear Unit**: $g(z) = max\{\alpha * z, z\}$. It overcomes the zero gradient issue from ReLU and assigns α which is a small value for $z \leq 0$.

If you're not sure which activation function to choose, start with ReLU. Next, we'll implement the above activation functions and draw a graph for each one to make it easier to see the domain and range of each function.

Feed Forward

Given its inputs from previous layer, each unit computes affine transformation $z = W^Tx + b$ and then apply an activation function $g(z)$ such as ReLU element-wise. During the process, we'll store (cache) all variables computed and used on each layer to be used in back-propagation. We'll write first two helper functions that will be used in the L-model forward propagation to make it easier to debug. Keep in mind that on each layer, we may have different activation function.

Cost

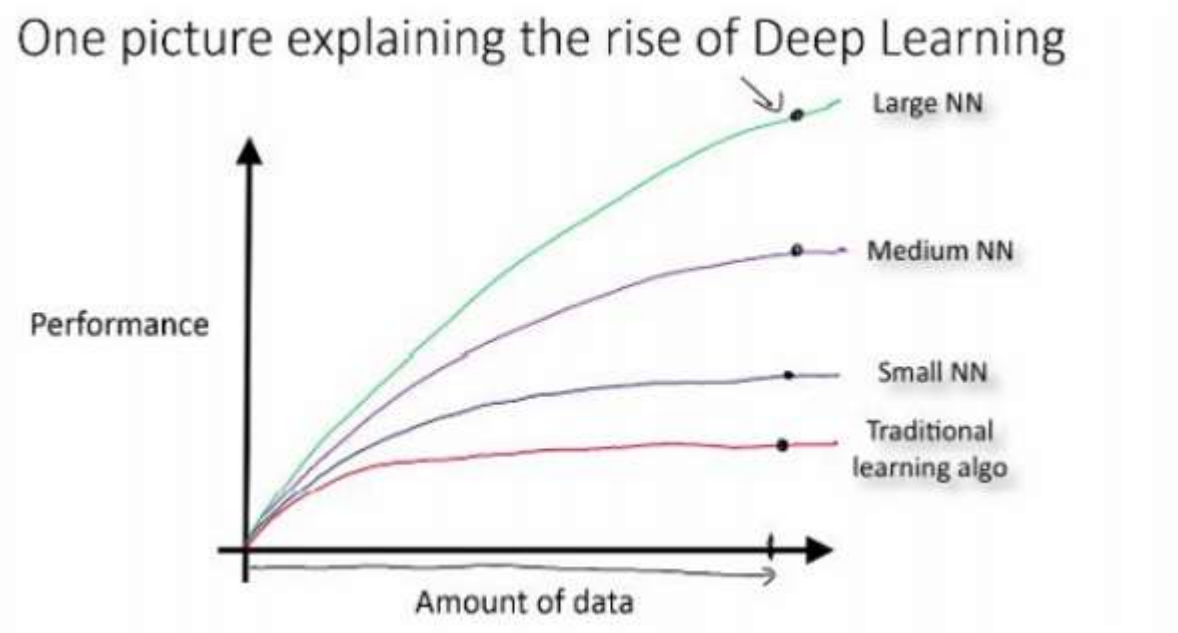
We'll use the binary **Cross-Entropy** cost. It uses the log-likelihood method to estimate its error. The cost is: The above cost function is convex; however, neural network usually stuck on a local minimum and is not guaranteed to find the optimal parameters. We'll use here gradient-based learning.

Back-Propagation

Allows the information to go back from the cost backward through the network in order to compute the gradient. Therefore, loop over the nodes starting at the final node in reverse topological order to compute the derivative of the final node output with respect to each edge's node tail. Doing so will help us know who is responsible for the most error and change the parameters in that direction. The following derivatives' formulas will help us write the back-propagate functions: Since b^l is always a vector, the sum would be across rows.

Deep Neural Networks:

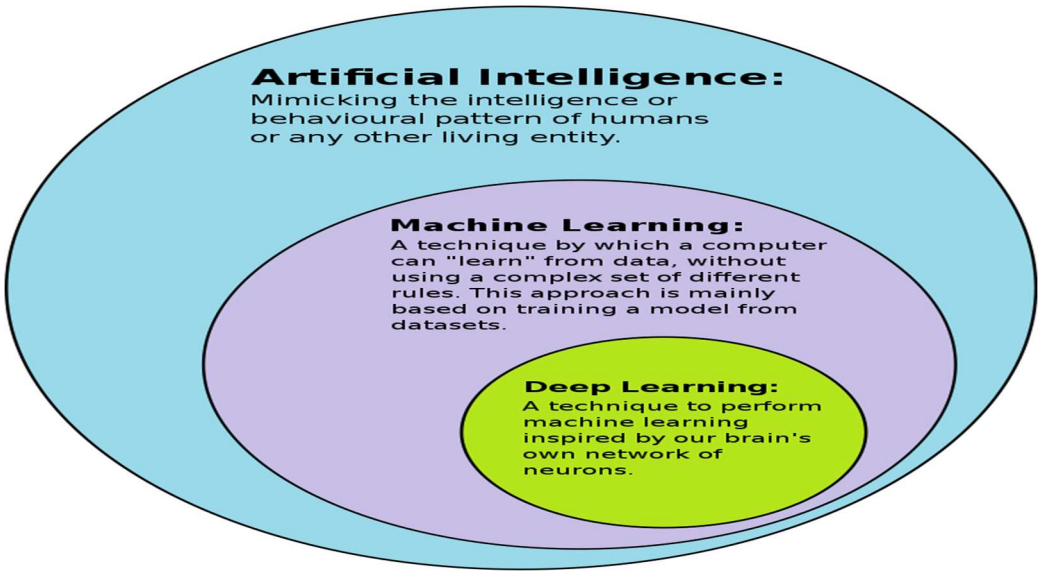
Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**

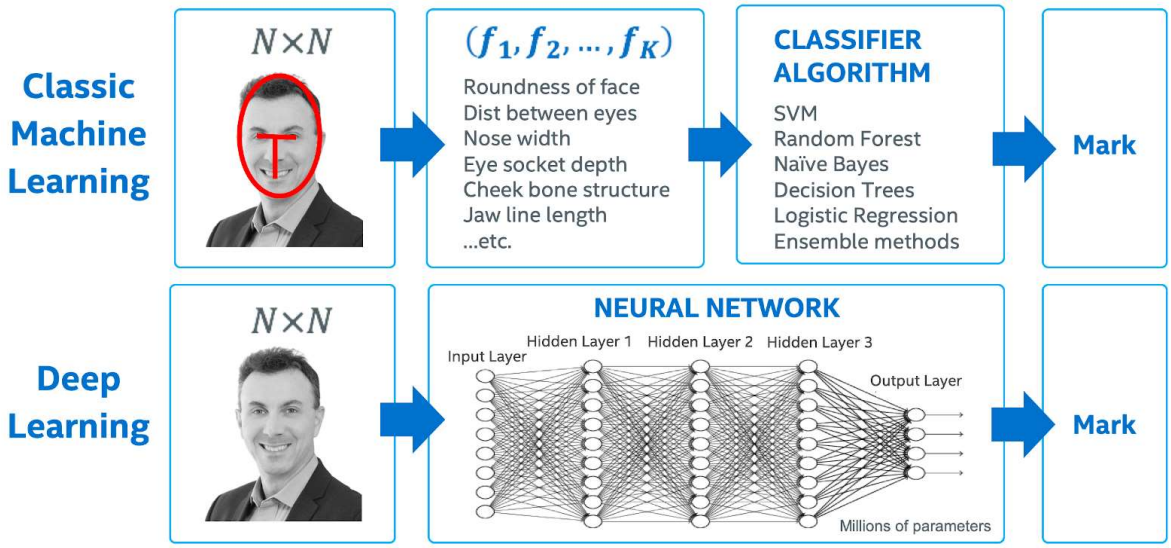


Deep-learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their **depth**; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as “deep” learning. So *deep* is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven’t heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer’s output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.





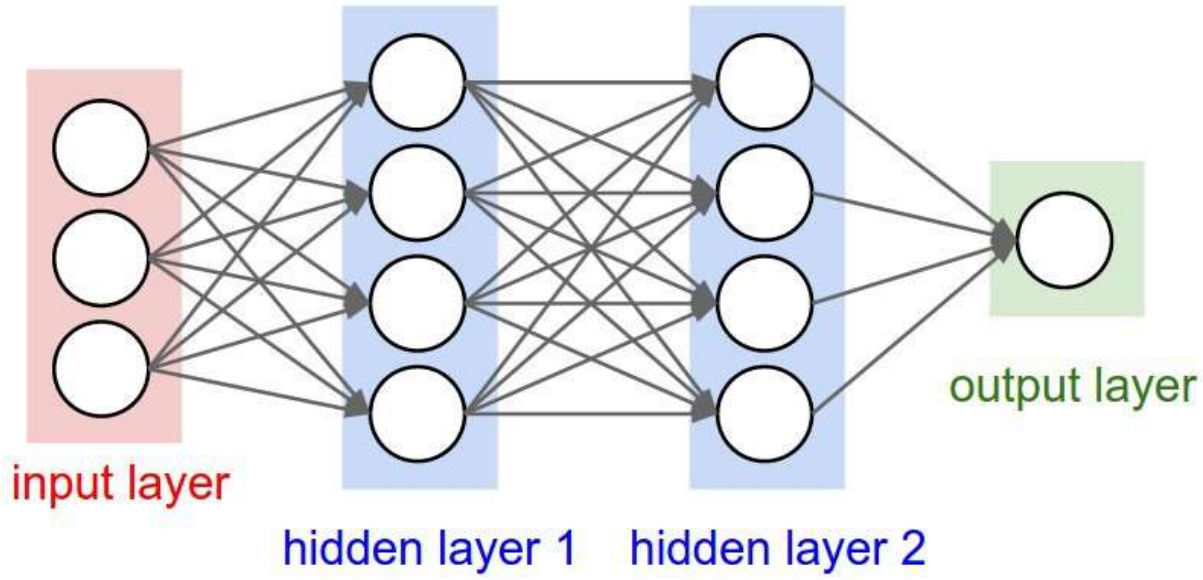
Working of Deep Learning

Deep learning algorithms utilizes supervised and unsupervised learning algorithms to train the outputs through the delivered inputs.

See the image below, these circles represent neurons that are interconnected. The neurons are classified into three different hierarchies of layers termed as Input, Hidden and Output Layers.

- The first neuron layer i.e. input layer receives the input data and passes it to the first hidden layer.

- The hidden layers perform the computations on the received data. The biggest challenge under neural networks creation is to decide the number of neurons and a number of hidden layers.
- Finally, the output layer produces the required output.



The popular algorithms that have been utilized to create a strong foundation for deep learning algorithms are:

1. Artificial Neural Network (ANN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Deep Neural Network (DNN)
5. Deep Belief Network (DBN)
6. Back Propagation
7. Stochastic Gradient Descent

Concept of Data Science:

Data Science Definition:

Data Science is the field that helps in extracting meaningful insights from data using programming skills, domain knowledge, and mathematical and statistical knowledge.

It helps to analyze the raw data and find the hidden patterns.

Basic Concepts that a Data Scientist should know:

1. Descriptive Statistics: Descriptive statistics help to analyze the raw data to find the primary and necessary features from it. Descriptive statistics offers a way to visualize the data to present it in a readable and meaningful way.
2. Probability: Probability is the mathematical branch that determines the likelihood of occurrence of any event in a random experiment. There are different types of probability, depending upon the type of event. Independent events are the two or more occurrences of an event that are independent of each other. Conditional probability is the probability of occurrence of any event having a relationship with any other event.
3. Dimensionality Reduction: Dimensionality reduction means reducing the dimensions of a data set so that it resolves many problems that do not exist in the lower dimension data.
4. Central Tendency: The central tendency of a data set is a single value that describes the complete data by the identification of a central value. There are different ways to measure the central tendency:
    - **Mean:** It is the average value of the data set column.
    - **Median:** It is the central value in the ordered data set.
    - **Mode:** The value repeating most in the data set column.
    - **Skewness:** It measures the symmetry of data distribution and determines if there is a long tail on either or both sides of the normal distribution.

- **Kurtosis:** It defines whether the data has a normal distribution or has tails.

5. Hypothesis Testing: Hypothesis testing is to test the result of a survey. There are two types of hypothesis as part of hypothesis testing viz. Null hypothesis and Alternate Hypothesis. The null hypothesis is the general statement that has no relation to the surveyed phenomenon. The Alternate hypothesis is the contradictory statement of the Null hypothesis.
6. Tests of Significance: Test of significance is a set of tests that helps to test the validity of the cited Hypothesis. Below are some of the tests that help in the acceptance or rejection of the Null Hypothesis.
    - **P-value test:** It is the probability value that helps to prove that the null hypothesis is correct or not. If p-value > a, then the Null Hypothesis is correct. If p-value < a, then the Null Hypothesis is False, and we reject it. Here 'a' is some significant value which is almost equal to 0.5.
    - **Z-Test:** Z-test is another way of testing the Null Hypothesis statement. It is used when the mean of two populations is different, and either their variances are known, or the size of the sample is large.
    - **T-test:** A t-test is a statistical test that is performed when either the variance of the population is not known or when the size of the sample is small.
7. Sampling Theory: Sampling is the part of statistics that involves the data collection, data analysis, and data interpretation of the data which is collected from a random set of population.
8. Bayesian Statistics: It is the statistical method that is based on the Bayes Theorem. Bayes theorem defines the probability of occurrence of an event depending upon the prior condition related to an event.
9. Machine Learning and Data Modeling: Machine learning is training the machine based on a specific data set with the help of a model. This trained model then makes future predictions. There are two types of machine learning modeling, i.e., supervised and

unsupervised. The supervised learning works on structured data where we predict the target variable. The unsupervised machine learning works on unstructured data that has no target field.

- **Machine Learning:** Machine learning is the subset of artificial intelligence where the machine learns from the previous experience and uses that to make predictions for the future.
- **Machine Learning Model:** A Machine Learning model is built to train the machine using some mathematical representation which then makes predictions.

10. Libraries in Data Science:

- **NumPy:** It is the basic library used for numerical computations. It is mainly used for data analysis.
- **Pandas:** It is the must-know library which is used for data cleaning, data storage, and time series.
- **SciPy:** It is another python library which is used to solve differential equations and linear algebra.
- **Matplotlib:** It is the data visualization library used to analyze correlation, determine outliers using scatter plot, and to visualize data distribution.
- **Scikit-Learn:** It is used to implement supervised and unsupervised machine learning models.

Traits of Big Data:

Big Data: Big Data is important because it enables organizations to gather store, manage and manipulate vast amounts of data at the right speed at the right time to gain the right insights.

Big Data is first introduced by Roger Magoulas to define amount of data that traditional data management techniques cannot manage and process due to complexity and size of this data.

Big Data refers to the huge collections of data that are structured and unstructured. This data may be sourced from servers, customer profile information, order and purchase data, financial transactions, ledgers, search history, and employee records. In large companies, this data collection is continuously growing with time.

Types of Big Data:

Big Data is present in three basic forms.

1. Structure Data: This kind of data is structured and is well-defined. It has a consistent order that can be easily understood by a computer or a human. This data can be stored, analyzed, and processed using a fixed format. The kind of data in databases, where it is neatly stored in columns and rows. Two sources of structured data are:
    a. **Machine-generated data** – This data is produced by machines such as sensors, network servers, weblogs, GPS, etc.
    b. **Human-generated data** – This type of data is entered by the user in their system, such as personal details, passwords, documents, etc. A search made by the user, items browsed online, and games played are all human-generated information.
2. Unstructured Data: The data that is not structured or well-defined is called unstructured data. This kind of data is unorganized and difficult to handle, understand and analyze.

    For example, unstructured data are your comments, tweets, shares, posts, and likes on social media. The videos you watch on YouTube and text messages you send via WhatsApp all pile up as a huge heap of unstructured data.
3. Semi Structured Data: This kind of data is somewhat structured but not completely. This may seem to be unstructured at first and does not obey any formal structures of data models such as RDBMS. For example, NoSQL documents have keywords that are used to process the document.
    CSV files are also considered semi-structured data.

Characteristics of Big Data:

The primary characteristics of Big Data are

1. Volume:

Volume refers to the huge amounts of data that is collected and generated every second in large organizations. This data is generated

from different sources such as IoT devices, social media, videos, financial transactions, and customer logs.
Storing and processing this huge amount of data was a problem earlier.

2. Variety:

   It refers to the different sources of data and their nature. The sources of data have changed over the years. Earlier, it was only available in spreadsheets and databases. Nowadays, data is present in photos, audio files, videos, text files, and PDFs. The variety of data is crucial for its storage and analysis.

3. Velocity:

   This term refers to the speed at which the data is created or generated. This speed of data producing is also related to how fast this data is going to be processed. This is because only after analysis and processing, the data can meet the demands of the clients/users.

   Massive amounts of data are produced from sensors, social media sites, and application logs – and all of it is continuous. If the data flow is not continuous, there is no point in investing time or effort on it.

4. Value:

   No matter how fast the data is produced or its amount, it has to be reliable and useful. Otherwise, the data is not good enough for processing or analysis. Research says that poor quality data can lead to almost a 20% loss in a company's revenue.

   Data scientists first convert raw data into information. Then this data set is cleaned to retrieve the most useful data. Analysis and pattern identification is done on this data set. If the process is a success, the data can be considered to be valuable.

5. Veracity

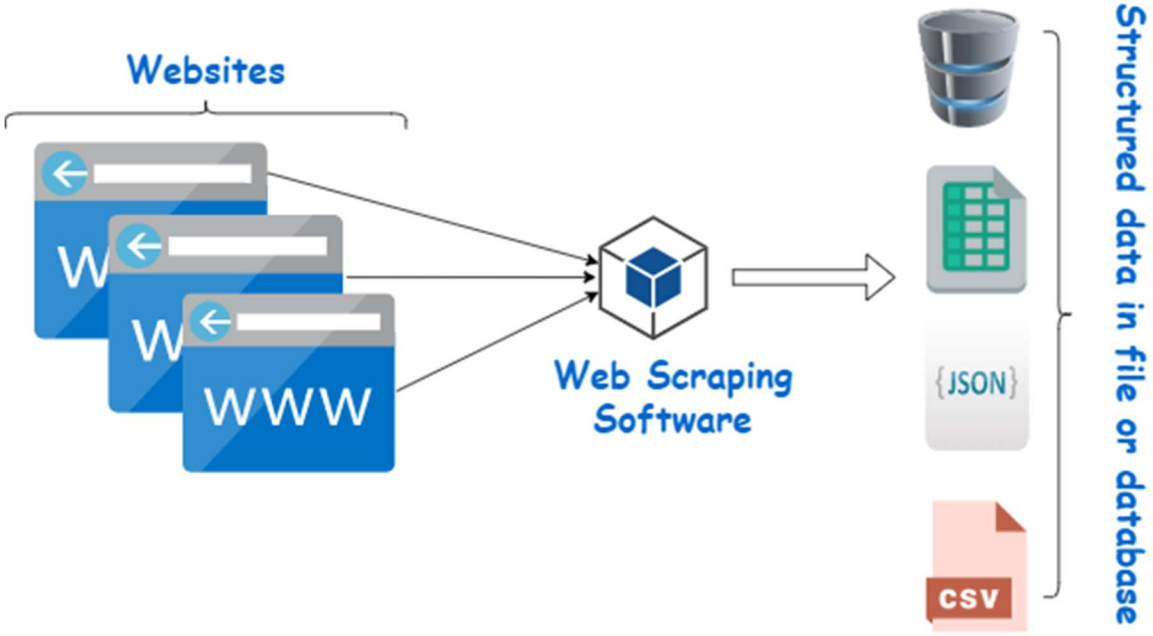
It defines the degree of trustworthiness of the data. As most of the data you encounter is unstructured, it is important to filter out the unnecessary information and use the rest for processing.

Web Scraping:

Web scraping is the process of using bots to extract content and data from a website. Web scraping extracts underlying HTML code and,

with it, data stored in a database. The scraper can then replicate entire website content.

When a web scraper needs to scrape a site, first it is provided the URL's of the required sites. Then it loads all the HTML code for those sites and a more advanced scraper might even extract all the CSS and Javascript elements as well. Then the scraper obtains the required data from this HTML code and outputs this data in the format specified by the user. Mostly, this is in the form of an Excel spreadsheet or a CSV file but the data can also be saved in other formats such as a JSON file.



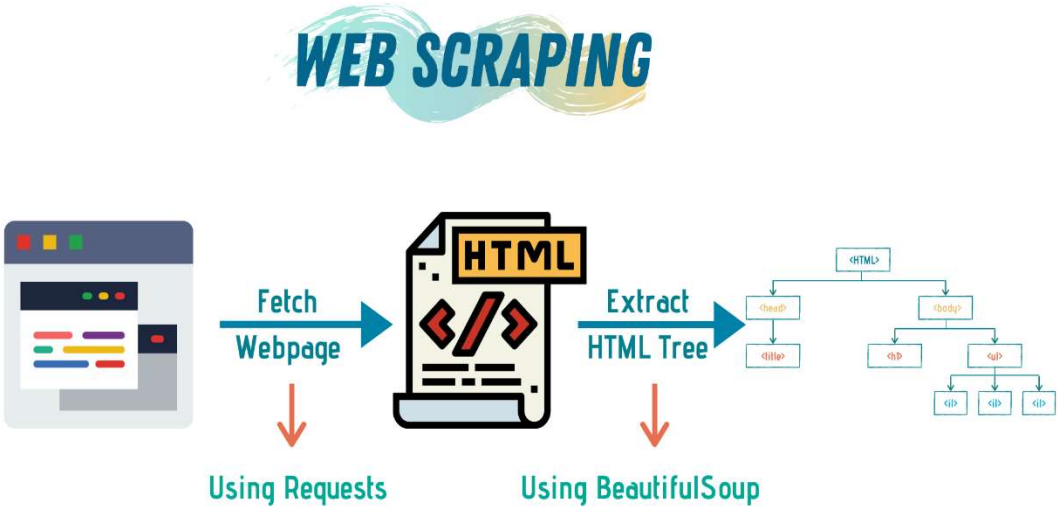
To extract data using web scraping with python, you need to follow these basic steps:

1. Find the URL that you want to scrape
2. Inspecting the Page
3. Find the data you want to extract
4. Write the code
5. Run the code and extract the data
6. Store the data in the required format

Web Scraping is the technique used by programmers to automate the process of finding and extracting data from the internet within a relatively short time.

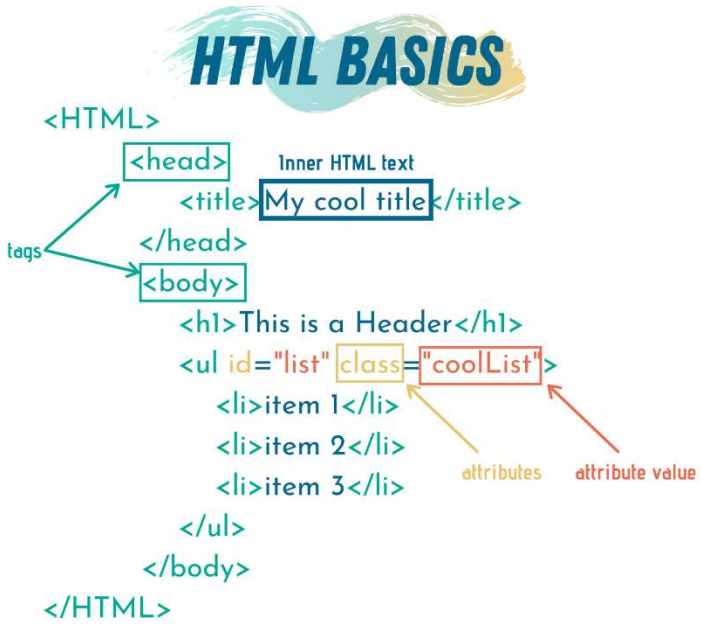
I will use *Python*, *Requests*, and *BeautifulSoup* to scrap some pages

To scrap and extract any information from the internet, follow three stages: Fetching HTML, Obtaining HTML Tree, then Extracting information from the tree.



the Requests library to fetch the HTML code from a specific URL. Then, we will use BeautifulSoup to Parse and Extract the HTML tree, and finally, we will use pure Python to organize the data.

In HTML, each opened tag must be closed. So, at the end of the HTML file, we need a closure tag </HTML>.



BeautifulSoup doesn’t fetch HTML from the web, it is, however, extremely good at extracting information from an HTML string.

```
some_html_str = """
<HTML>
   <HEAD>
      <TITLE>My cool title</TITLE>
   </HEAD><BODY>
   <H1>This is a Header</H1>
   <ul id="list" class="coolList">
      <li>item 1</li>
      <li>item 2</li>
      <li>item 3</li>
   </ul>
</BODY>
</HTML>
"""
#Feed the HTML to BeautifulSoup
soup = bs(some_html_str)
```

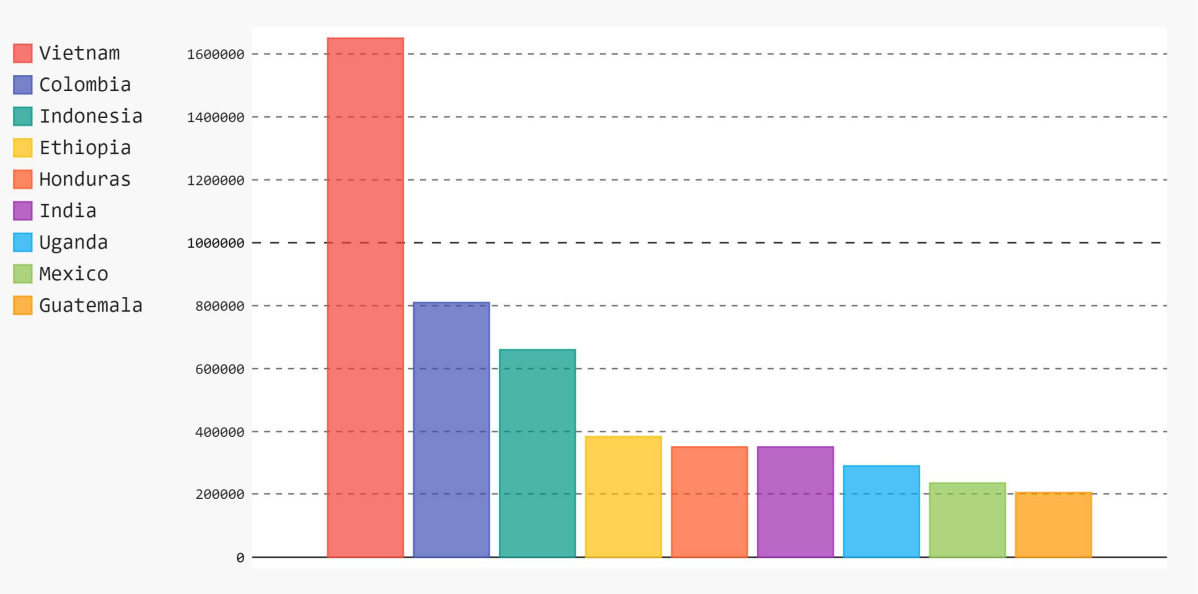
The variable `soup` now has the information extracted from the HTML string. We can use this variable to obtain information from the HTML tree.

BeautifulSoup has many functions that can be used to extract specific aspects of the HTML string. However, two functions are used to most: `find` and `find_all`.
The function `find` returns only the first occurrence of the search query, while `find_all` returns a list of all matches.

To fetch the HTML we will use the Requests library and then pass the fetched HTML to BeautifulSoup.

```
#Import needed libraries
from bs4 import BeautifulSoup as bs
```

```
import requests as rq
import pygal
from IPython.display import display, HTML
#Fetch HTML
url = 'https://en.wikipedia.org/wiki/List_of_countries_by_coffee_production'
#Extract HTMl tree
page = rq.get(url).text
soup = bs(page)
#Find countries and quantiy
table = soup.find('table')
top_10_countries = []
for row in table.find_all('tr')[2:11]:
    temp = row.text.replace('\n\n',' ').strip() #obtain only the quantiy in tons
    temp_list = temp.split()
    top_10_countries.append((temp_list[0],temp_list[2]))
#Plot the top 10 countries
bar_chart = pygal.Bar(height=400)
[bar_chart.add(item[0],int(item[1].replace(',',''))) for item in top_10_countries]
display(HTML(base_html.format(rendered_chart=bar_chart.render(is_unicode=True))))
```



Tools for Scraping the Web page:

1. Beautiful Soup: **BeautifulSoup** is another Python library, commonly used to parse data from XML and HTML documents. Organizing this parsed content into more accessible trees,
2. Scrapy: **Scrapy** is a Python-based application framework that crawls and extracts structured data from the web.

Step #1: Fetch main URL HTML code

```
url = 'https://en.wikipedia.org/wiki/Time%27s_List_of_the_100_Best_Novels'
page = rq.get(url).text
```

Step #2: Feed that HTML to BeautifulSoup

```
soup = bs(page)
```

Step #3: Extract each book from the list and get the wiki link of each book

```
rows = soup.find('table').find_all('tr')[1:]
books_links = [row.find('a')['href'] for row in rows]
base_url = 'https://en.wikipedia.org'
books_urls = [base_url + link for link in books_links]
```

Step #4: Obtain data for each book

We will first consider only one book, assume it's the first one in the list. If we open the wiki page of the book we will see the different information of the book enclosed in a table on the right side of the screen. Going through the HTML we can see where everything is stored.

To make things easier and more efficient, I wrote custom functions to extract different information from the book's wiki.

Step #5: Get all books data, clean, and plot final results

We have all we need to automate the code and run it. I have to say, collecting data is not always a 100% accurate,

```
import requests as rq
import pygal
import time
import pygal
from bs4 import BeautifulSoup as bs
from IPython.display import display, HTML


base_html = """
<!DOCTYPE html>
<html>
  <head>
  <script                                          type="text/javascript"
src="http://kozea.github.com/pygal.js/javascripts/svg.jquery.js"></s
cript>
  <script                                          type="text/javascript"
src="https://kozea.github.io/pygal.js/2.0.x/pygal-
tooltips.min.js""></script>
  </head>
  <body>
    <figure>
      {rendered_chart}
    </figure>
  </body>
</html>
"""
#define functions for data collection
```

```
def find_book_name(table):
    if table.find('caption'):
        name = table.find('caption')
    return name.text


def get_author(table):
    author_name = table.find(text='Author').next.text
    return author_name


def get_genre(table):
    if table.find(text='Genre'):
        genre = table.find(text='Genre').next.text
    else:
        genre = table.find(text='Subject').next.next.next.text
    return genre


def get_publishing_date(table):
    if table.find(text='Publication date'):
        date = table.find(text='Publication date').next.text
    else:
        date = table.find(text='Published').next.text
    pattern = re.compile(r'\d{4}')
    year = re.findall(pattern, date)[0]
    return int(year)


def get_pages_count(table):
    pages = table.find(text='Pages').next.text
    return int(pages)


def parse_wiki_page(url):
    page = rq.get(url).text
    return bs(page)
```

```
def get_book_info_robust(book_url):
    #To avoid breaking the code
    try:
        book_soup = parse_wiki_page(book_url)
        book_table = book_soup.find('table',class_="infobox vcard")
    except:
        print(f"Cannot parse table: {book_url}")
        return None

    book_info = {}
    #get info with custom functions
    values = ['Author', 'Book Name', 'Genre',
          'Publication Year', 'Page Count']
    functions = [get_author, find_book_name, get_genre,
            get_publishing_date, get_pages_count]


    for val, func in zip(values, functions):
        try:
            book_info[val] = func(book_table)
        except:
            book_info[val] = None


    return book_info
#Get books
url                                                                    =
'https://en.wikipedia.org/wiki/Time%27s_List_of_the_100_Best_N
ovels'
page = rq.get(url).text
soup = bs(page)
rows = soup.find('table', class_="wikitable sortable").find_all('tr')[1:]
books_links = [row.find('a')['href'] for row in rows]
base_url = 'https://en.wikipedia.org'
books_urls = [base_url + link for link in books_links]
```

```
#to store books info
book_info_list = []
#loop first books
for link in books_urls:
  #get book info
  book_info = get_book_info_robust(link)
  #if everything is correct and no error occurs
  if book_info:
    book_info_list.append(book_info)
  #puase a second between each book
  time.sleep(1)

#Collect different genres
genres = {}
for book in book_info_list:
    book_gen = book['Genre']
    if book_gen:
        if 'fiction' in book_gen or 'Fiction' in book_gen:
            book_gen = 'fiction'
    if book_gen not in genres: #count books in each genre
        genres[book_gen] = 1
    else:
        genres[book_gen] += 1

print(genres)
#Plot results
bar_chart = pygal.Bar(height=400)
[bar_chart.add(k,v) for k,v in genres.items()]
display(HTML(base_html.format(rendered_chart=bar_chart.render(
is_unicode=True))))
```

Analysis and Reporting:

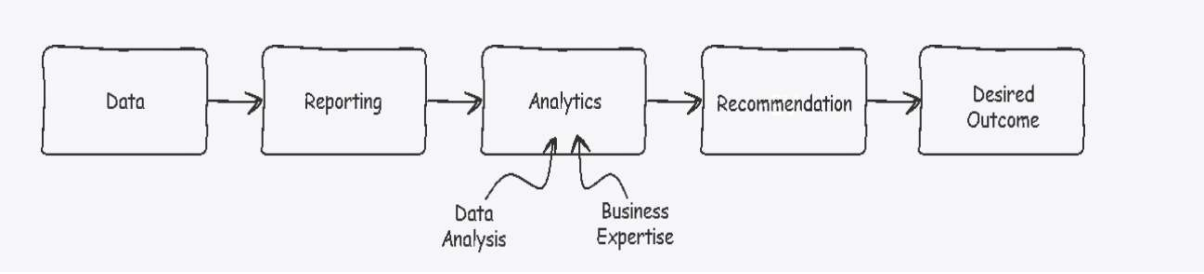
In a highly competitive world, there is an increasing demand for data to make decisions. Within established companies, this is evidenced through the availability of reports such as financial reports, accounting reports, market reports and many others.

“Reporting” means data to inform decisions. Typical reporting requests usually imply repeatable access to the information, which could be monthly, weekly, daily, or even real-time.

- **Data is available**: often data needs to be sourced from disparate source systems which are often fragmented within the companies or outside the companies
- **Data is clean**: often data needs to be translated for human consumption and needs to be shaped for analysis enablement.

Reporting extracts data from various data sources, allows comparisons, and makes the information easier to understand by summarizing and visualizing the data in tables, charts and dashboards.

Analytics drives business decisions by questioning and interpreting the data with a distinct purpose in mind. Any good data scientist will tell you that analytics without business purpose, intelligence, and expertise applied can be very dangerous, often leading to erroneous findings.



Reporting is definitely valuable in business—evidenced by the fact that you see it everywhere. Analytics are much harder to deliver. But the value is immense when effective data analysis, injected with business knowledge is combined to bring actionable recommendations and desired outcomes.

Structure of a Data Analysis Report:

The overall structure of a data analysis report is simple:

1. Introduction

2. Body

3. Conclusion(s)/Discussion

4. Appendix/Appendice

1. Introduction.

Good features for the Introduction include:

• Summary of the study and data, as well as any relevant substantive context, background, or framing issues.

• The "big questions" answered by your data analyses, and summaries of your conclusions about these questions.

• Brief outline of remainder of paper. The above is a pretty good order to present this material in as well.

2. Body. The body can be organized in several ways. Here are two that often work well:

• Traditional. Divide the body up into several sections at the same level as the Introduction, with names like: – Data – Methods – Analysis – Results This format is very familiar to those who have written psych research papers. It often works well for a data analysis paper as well, though one problem with it is that the Methods section often sounds like a bit of a stretch: In a psych research paper the Methods section describes what you did to get your data. In a data analysis paper, you should describe the analyses that you performed. Without the results as well, this can be pretty sterile sounding, so I often merge these "methods" pieces into the "Analysis" section when I write.

• Question-oriented. In this format there is a single Body section, usually called “Analysis”, and then there is a subsection for each question raised in the introduction, usually taken in the same order as in the introduction (general to specific, decreasing order of importance, etc.). Within each subsection, statistical method, analyses, and conclusion would be described (for each question).

For example:

3. Analysis

3.1 Success Rate Methods Analysis Conclusions

3.2 Time to Relapse Methods Analysis Conclusions

3.3 Effect of Gender Methods Analysis Conclusions

3.4 Hospital Effects Methods Analysis Conclusions Etc. .

Other organizational formats are possible too. Whatever the format, it is useful to provide one or two well-chosen tables or graphs per question in the body of the report, for two reasons: First, graphical and tabular displays can convey your points more efficiently than words; and second, your “skimming” audiences will be more likely to have their eye caught by an interesting graph or table than by running text. However, too much graphical/tabular material will break up the flow of the text and become distracting; so extras should be moved to the Appendix.

3. Conclusion(s)/Discussion. The conclusion should reprise the questions and conclusions of the introduction, perhaps augmented by some additional observations or details gleaned from the analysis section. New questions, future work, etc., can also be raised here.

4. Appendix/Appendices. One or more appendices are the place to out details and ancillary materials.

These might include such items as

• Technical descriptions of (unusual) statistical procedures

• Detailed tables or computer output

• Figures that were not central to the arguments presented in the body of the report

• Computer code used to obtain results. In all cases, and especially in the case of computer code, it is a good idea to add some text sentences as comments or annotations, to make it easier for the uninitiated reader to follow what you are doing.

It is often difficult to find the right balance between what to put in the appendix and what to put in the body of the paper. Generally you should put just enough in the body to make the point, and refer the reader to specific sections or page numbers in the appendix for additional graphs, tables and other details.

Data Science in Business:

Data Science for Business teaches you how to effectively use data to tackle your business decisions and motivate those around you to take action based on evidence.

Business Intelligence for Making Smarter Decisions:

1. Understanding the context and nature of the problem that we are required to solve.
2. Exploring and quantifying the quality of the data.
3. Implementation of the right algorithm and tools for finding a solution to the problems.
4. Using story-telling to translate our insights for a better understanding of teams.

Making Better Products:

Companies should be able to attract their customers towards products. They need to develop products that suit the requirements of customers and provide them with guaranteed satisfaction. Therefore, industries require data to develop their product in the best possible way.

The process involves the **analysis of customer reviews** to find the best fit for the products.

Managing Businesses Efficiently:

Data Science platforms unearth the hidden patterns that are present inside the data and help to make meaningful analysis and prediction of events.

With Data Science, businesses can manage themselves more efficiently. Both large scale businesses and small startups can benefit from data science in order to grow further.

Data Scientists help to analyze the health of the businesses. With data science, companies can predict the success rate of their strategies. Data Scientists are responsible for turning raw data into cooked data.
This helps in summarizing the performance of the company and the health of the product. Data Science identifies key metrics that are essential for the determination of business performance.

Based on this, the business can take important measures to quantify and evaluate its performance and take appropriate management steps. It can also help the managers to analyze and determine the potential candidates for the business.

Using data science, businesses can also foster leadership development by tracking the performance, success rate, and other important metrics. With workforce analytics, industries can evaluate what is best working for the employees.

Predictive Analytics to Predict Outcomes

Predictive analytics is the most important part of businesses. With the advent of advanced predictive tools and technologies, companies have expanded their capability to deal with diverse forms of data.

In formal terms, predictive analytics is the **statistical analysis of data** that involves several machine learning algorithms for predicting the future outcome using the historical data.

Leveraging Data for Business Decisions

we understood how data science is playing an important role in predicting the future. These predictions are necessary for businesses to

learn about future outcomes. Based on this, businesses take decisions that are data-driven.

Assessing Business Decisions

After making decisions through the forecast of the future occurrences, it is a requirement for the companies to assess them. This is possible through several hypothesis testing tools.

After implementing the decisions, businesses should understand how these decisions affect their performance and growth. If the decision leads to any negative factor, then they should analyze it and eliminate the problem that is slowing down their performance.
There are various procedures through which businesses can evaluate their decisions and plan a suitable action strategy. These decisions revolve around their customer requirements, company goals as well as the needs of the project executives.

Automating Recruitment Processes

Data Science has played a key role in bringing automation to several industries. It has taken away the mundane and repetitive jobs. One such job is that of resume screening. Every day, companies have to deal with hordes of applicant's resumes.

Some major businesses can even attract thousands of resumes for a position. In order to make sense of all of these resumes and select the right candidate, businesses make use of data science.

The data science technologies like image recognition are able to convert the visual information from the resume into a digital format. It then processes the data using various **analytical algorithms like clustering and classification** to churn out the right candidate for the job.

Data science is on a continued upswing — both in terms of career opportunities as well as in the ways that organizations, across industries, are making use of it. Everyone from HDFC bank and Flipkart to the government of India is leveraging data science platforms, methods and techniques. Clearly, there's no time like the

present to build a data science career. And a great way to start is by developing skills in a few data science tools.

Our Springboard experts recommend the top 15 data science tools to learn in 2020. Don't rush to learn them all. Work on becoming conversant in as many as you can; but get hands-on experience in one or two by experimenting with them on your data science projects.

Top Data Science Tools to Learn in 2020

Data Mining and Transformation

Data mining is about identifying patterns in large datasets. But in practice, it has come to include extraction, collection, storage and analysis of information. There are tools that can do one or more of these tasks. Our top three are:

1. Weka is a popular tool used for data mining, pre-processing and classifying data. Weka's GUI simplifies classification, association, regression, and clustering, providing statistically robust results.
2. Scrapy is best for writing web spiders that will crawl websites and extract data. Written in Python, Scrapy is fast and powerful. CareerBuilder uses Scrapy to collect data on job offers across multiple sites.
3. Pandas is a popular data wrangling software, also written in Python. It is perfect for working with numerical tables and time-series data. It provides flexible data structures that make data manipulation easy. It is the backbone of recommendation engines from Netflix and Spotify.

Data Analysis and Big Data Tools

Once the data has been collected and processed, it's time for analysis. Here, you need a tool to get the data ready for model training and refining predictions. Some of the best ones are:

1. KNIME, or Konstanz Information Miner, in full, provides end-to-end data analysis, and integration and reporting. Its GUI allows

users to perform pre-processing, analysis, model building and visualization with minimal programming.
2. Hadoop is a software framework primarily used for storage and processing of big data on a distributed model. This allows the data to be processed faster, and any hardware failures to be handled better.
3. Spark from Apache is an analytics engine for big data. With Spark, you can run large-scale workloads of peta-bytes of data and build applications faster, and deploy them comfortably across virtual machines, containers, on-prem or on the cloud.
4. Neo4J is a graph database management platform, and the most popular one at that. Unlike a relational database, graph databases store connections along with the data, and Neo4J helps users detect hard-to-find patterns on such data.

Model Deployment

One of the key purposes of data science is in developing machine learning models on the data. These models can be logical, geometric or probabilistic models. Here are some tools you can use for model-building.

1. TensorFlow.js is the JavaScript edition of the popular machine learning framework, TensorFlow. You can develop models in JavaScript or Node.js and use TensorFlow.js to deploy them over the web on the client browser.
2. MLFlow is a machine learning lifecycle management platform — from building and packaging to deploying models. If you're experimenting with multiple tools or building several models, MLFlow helps manage all of it from one place. You can integrate library, language or algorithm with the product.

Data Visualization

Data visualization needs to be more than just a visual representation of data. Today, it needs to be scientific, visual and more importantly insightful. In that, it should go beyond reporting; it must present analytical reasoning through interactive visual interfaces. Here are some tools that can help visualize your data science projects.

1. Orange is an easy to use data visualization tool with a large toolkit. In spite of being a GUI-based beginner-friendly tool, you mustn't mistake it for a light-weight one. It can do statistical distributions and box plots as well as decision trees, hierarchical clustering and linear projections.
2. With D3.js or Data-Driven Documents (D3), you can visualize data on web browsers using HTML, SVG and CSS. It is popular with data scientists for its capabilities in animation and interactive visuals.
3. Ggplot2 helps data scientists create aesthetically pleasing and elegant visualizations, using R. So next time you want to really wow your audience, you know which library to choose for creating your visuals!

Development Environments

Like most programming, writing and deploying data science code can also be done more efficiently with an integrated development environment. IDEs offer code insights, test your code, help you identify errors easily, and even allow you to run your code directly with plugins. Here are some IDEs especially for data science-related code.

1. Jupyter Notebooks is a web application that can host code, data, notes, equations, etc. — in other words, an interactive online document. If you're working on a project with other data scientists, Jupyter Notebooks is the perfect collaboration tool!
2. Zeppelin Notebooks is a web-based environment where you can perform data analysis using many languages like Python, SQL, Scala etc. You can explore, share, analyze, and visualize data with Zeppelin Notebooks.
3. R Studio's biggest attraction is that it integrates R-based tools into a single environment. You can write clean code, execute it, manage workflows and even debug it with R Studio.