**DATA PREPROCESSING**

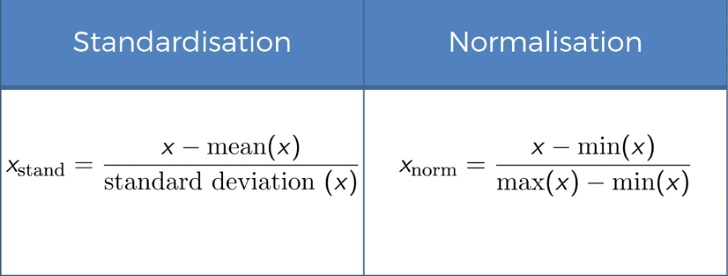
1. **Missing Data:** from sklearn.preprocessing import Imputer  
    fill the missing data with mean or mode or relevent feature provided
2. **Encoding Categorical Variables:** the column which contain only categorical variable are called categorical variable and machine learning models are based on mathematical equation so categorical data would create problem if keep text of categorical data in equation so we only wants numbers in the equations.  
   So we need to encode categorical variable into numbers.

**Python:** from sklearn.preprocessing library import LabelEncoder class

Problem in labeling: if we are labeling categorical data like countries and we give them labels like 0, 1 and so on so equation will understand that one country value is greater than others which is not good. LabelsEncoder only encodes the value without bothering if there is order or not.  
**OneHotEncoder:** converts categorical value to numric value such that equation doesn’t priortise with other.

1. **Feature scaling:** why do we need that? Because many ML model works on the principle of Eculidean distance and if the scale are not same than the paramter having high numerical value will be given more importance than other and other will be neglected which is not good.

**Note**: Every model in sci-kit uses feature scaling so we don’t have to provide specifically.  
  
Types of Feature scaling:



We are putting our variable in the same range in same scale in above method.

# feature scaling

from sklearn.preprocessing import StandardScalar

sc\_x = StandardScalar()

x\_train = sc\_x.fit\_transform(x\_train)

x\_test = sc\_x.transform(x\_test)

Do we need to scale dummy variable?

It depends on the context but if we don’t scale here it wont affect as our feature variable are between (-1, 1).

Note: Sometime even if ML model are not based on Eculidean distance than also feature scaling is important as algorithm will converge much faster. That will be the case for Decision tree.

**Note:** We don’t need to apply feature scaling on y (dependent variable) as it is classification problem, we have to in regression problem.

**Regression**

**Simple Linear Regression:**



Linear Regression being trend line that best fit our data.

**Multiple Linear regression:**



y = dependent variable

Assumption of Linear Regression:

1. Linearity
2. Homoscedasticity
3. Multivariate normality
4. Independence of error
5. Lack of multicollinearity

**Dummy variable trap**: 

We don’t include both variable as one variable is alternate of others in categorical value.

When we are building a model always emit one dummy variable.

**5 method of building models:**

1. All-in: just throw in all variables, avoid this
2. Backward Elimination:
   * 1. Select a significance level to stay in a model (eg. Sl(significance level) = 0.05)
     2. Fit the full model with all possible predictors
     3. Consider the predictor with the highest p-value. If p>sl, goto step 4, otherwise finish
     4. Remove the predictor
     5. Fit model without this variable

Ques. What is p value?

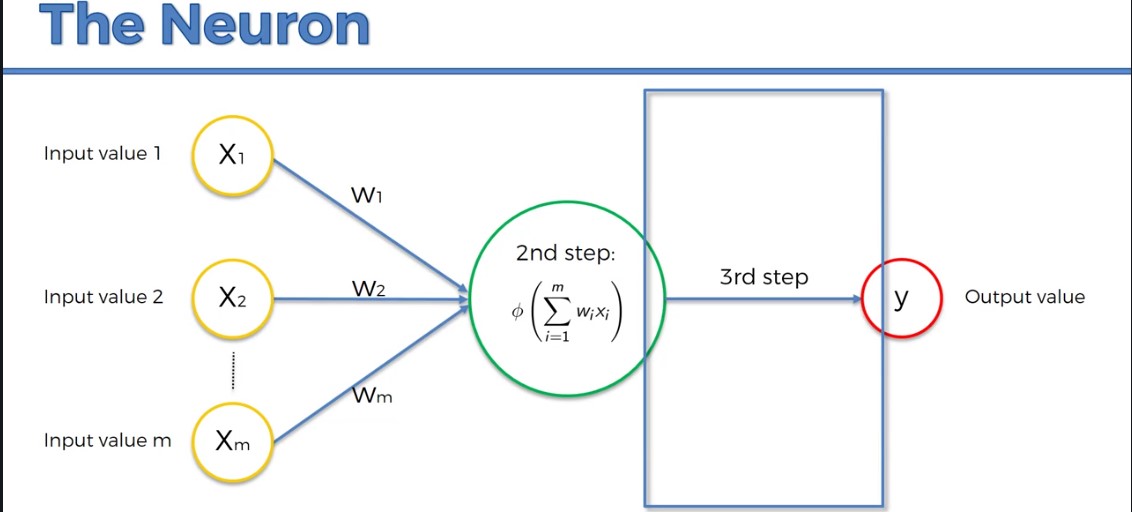
1. Forward selection:
   * 1. Select a significance level to enter the model (eg. Sl = 0.05)
     2. Fit all the simple regression model select the one with lowest p value
     3. Keep this variable and fit all possible models with one extra predictor added to the one(s) you already have
     4. Consider the predictor with the lowest p-value. If p<sl goto step 3, otherwise finish
     5. Finish: keep the previous model and this is our required model
2. Bidirectional Elimination
   * 1. Select a significance level to stay and significance level to enter eg. Slenter=0.05, slstay=0.05
     2. Perform the next step of forward selection(new variable must have p< slenter to enter)
     3. Perform all stepp of backward elimination(old variables must have p<slstay to stay)
     4. No new variable can enter and no new variable can exit
3. Score comparision
   * 1. Select a criterion of goodness of fit
     2. Construct all possible regression model:  total combinations
     3. Select the one with best criterion

**Natural Language Processing:**

* + 1. In NLP we should go for ‘tsv’ file instead of csv because people use comma as a general but not tab so tab as a seperator is good.
    2. The **bag-of-words** model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the **bag** (multiset) of its **words**, disregarding grammar and even **word** order but keeping multiplicity.
    3. Use stopword as set for large set of data for sentiment analysis like book or big document as set is faster than list  
       syntax: set(stopwords.words('english'))
    4. Stemming - process of keeping only root word for easier and effective sentiment analysis.

**Artificial Neural Network:**

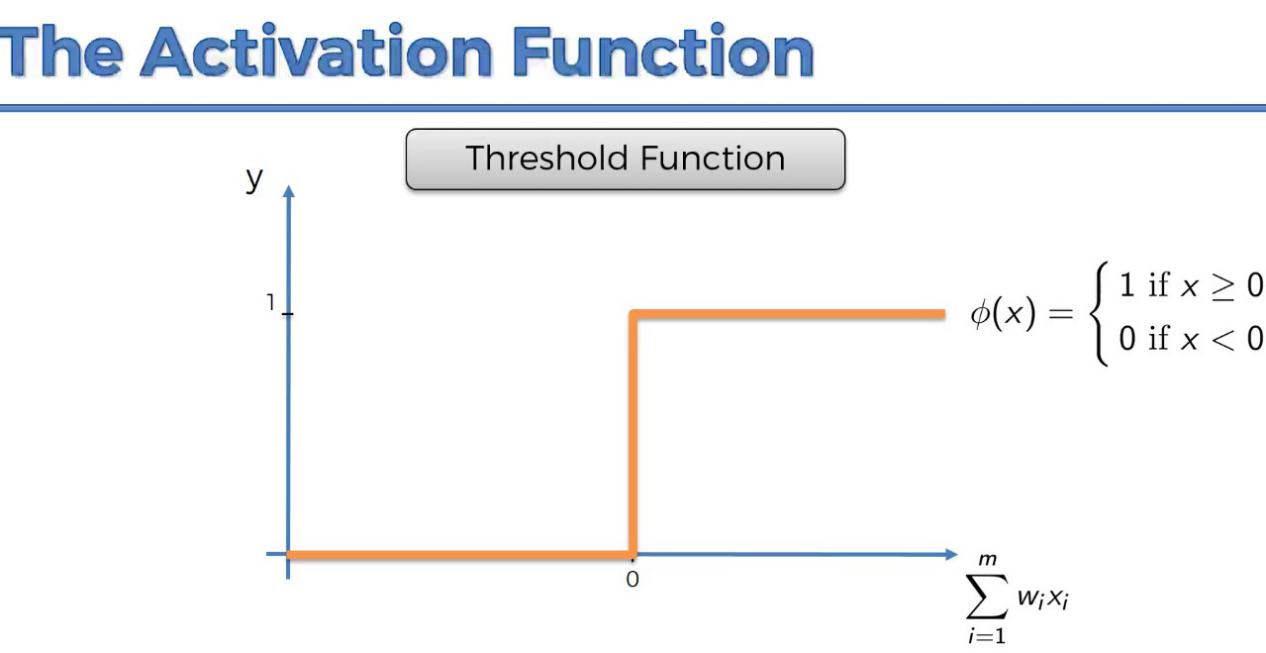
1. **Neuron:** Basic building block of artificial neural network.
2. **Weights:** are how ANN learns by adjusting the weight.



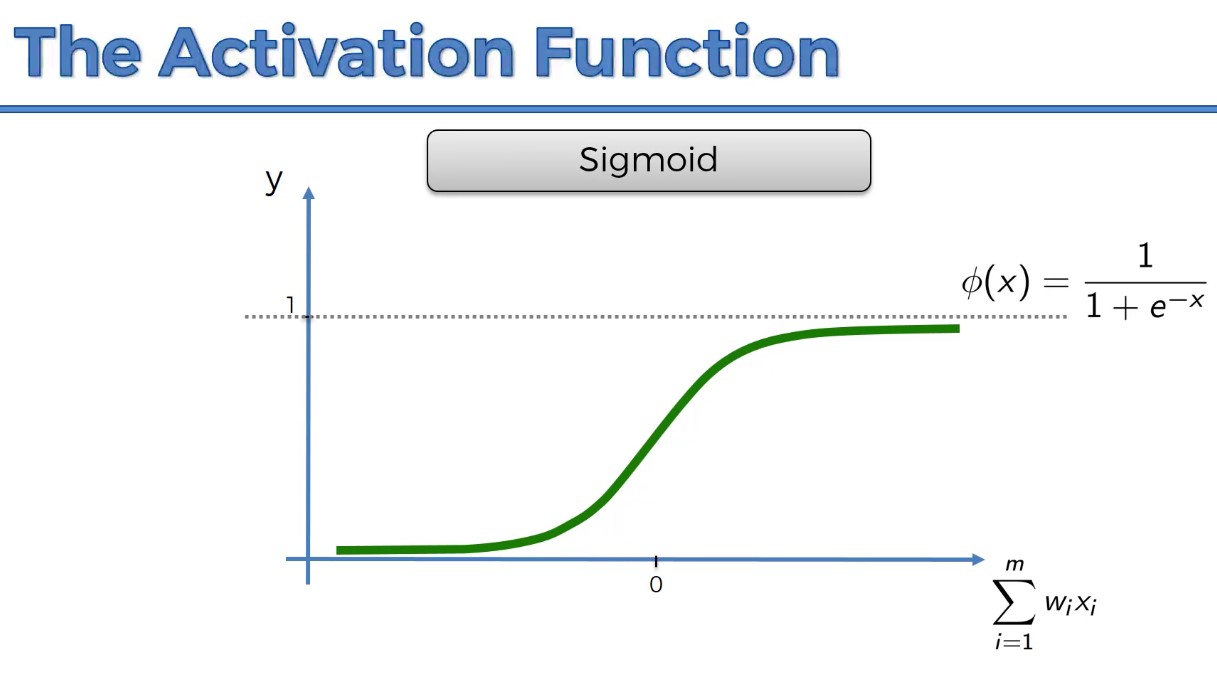
1. **The activation function:** In computational **networks**, the **activation function** of a node defines the output of that node given an input or set of inputs. ... In artificial **neural networks** this **function** is also called the transfer **function.**

There are 4 types of activation function:

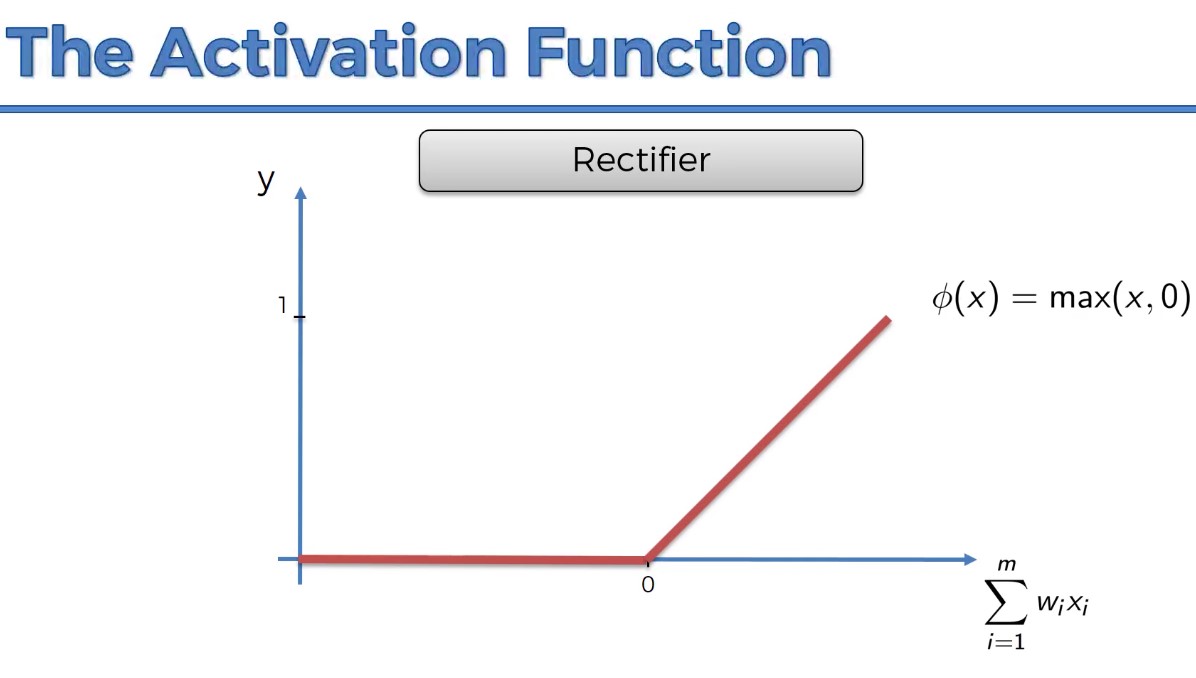
* 1. Threshold Function:



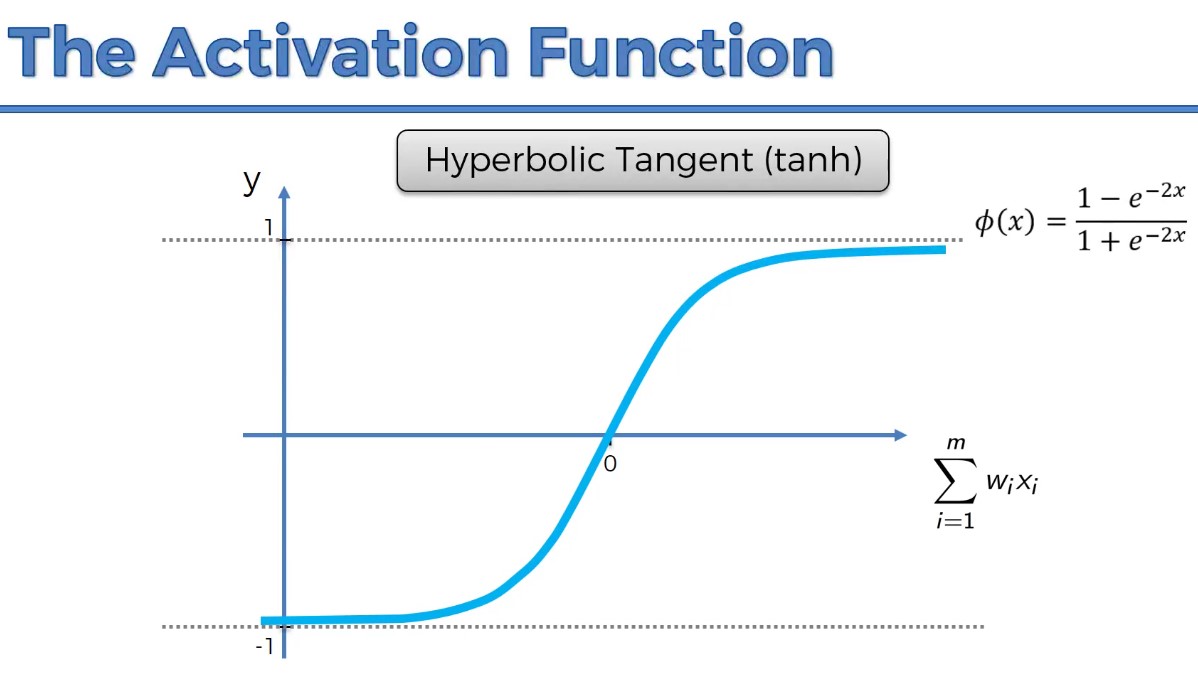
b) **Sigmoid Function:**



c) Rectifier Function:



d) Hyperbolic Function:



Note: Generally in the hidden layer we apply rectifier function and in the output layer we apply sigmoid function.

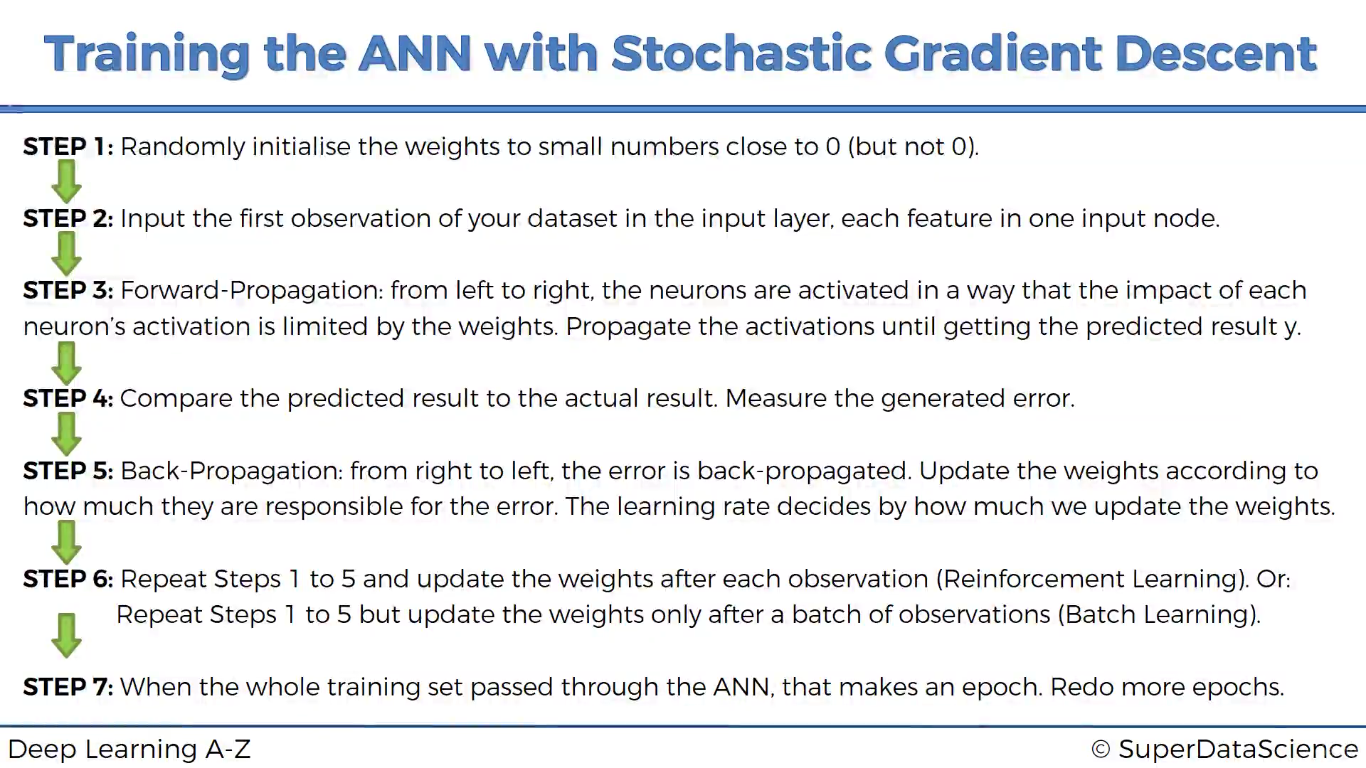
**Cost Function:** This tells how much the actual output differ from real output. We have to minimise cost function and we can achieve this by updating the weight.



In order for Neural Network to learn what needs to happen is **Back Propgation** and that is when the error the differences of sum of squared error is back propgated through the neural network and weights are adjusted accordingly.

**Gradient Descent:** Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost). Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

**Stochastic Gradient Descent:**Advantage over Batch gradient descent that:  
 a) Its finds the global minima of cost function rather than local minima as case may happen in Batch Gradient descent.  
b) It is much faster than batch GD.

**Backpropgation:  
**