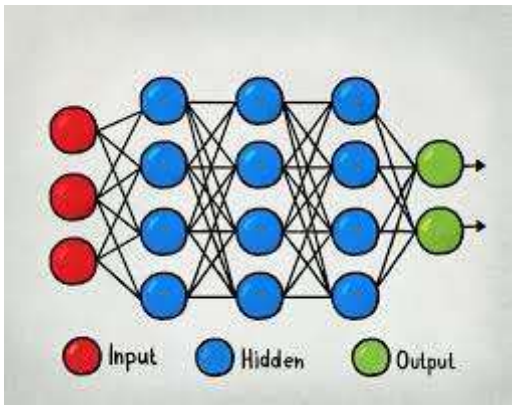
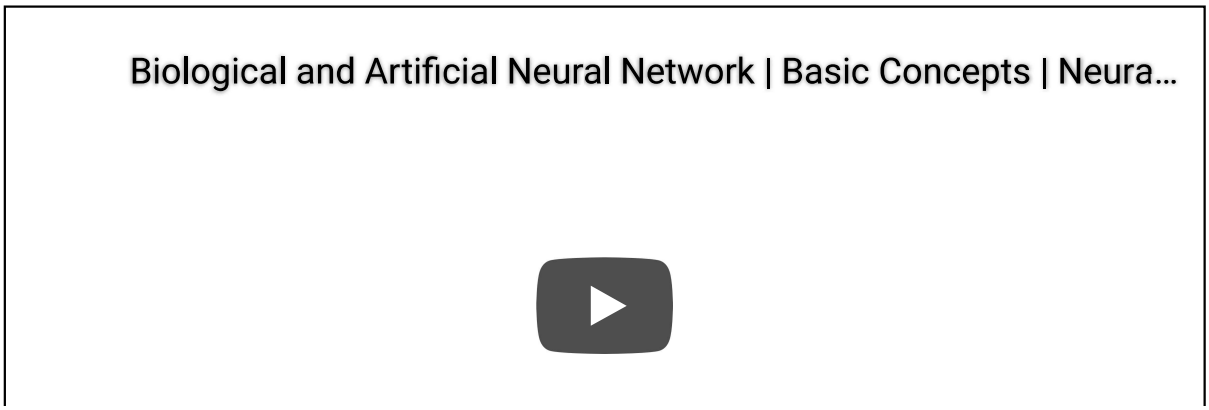
# Assignment-5: Artificial Neural Networks_ANN


## Welcome back guys!!

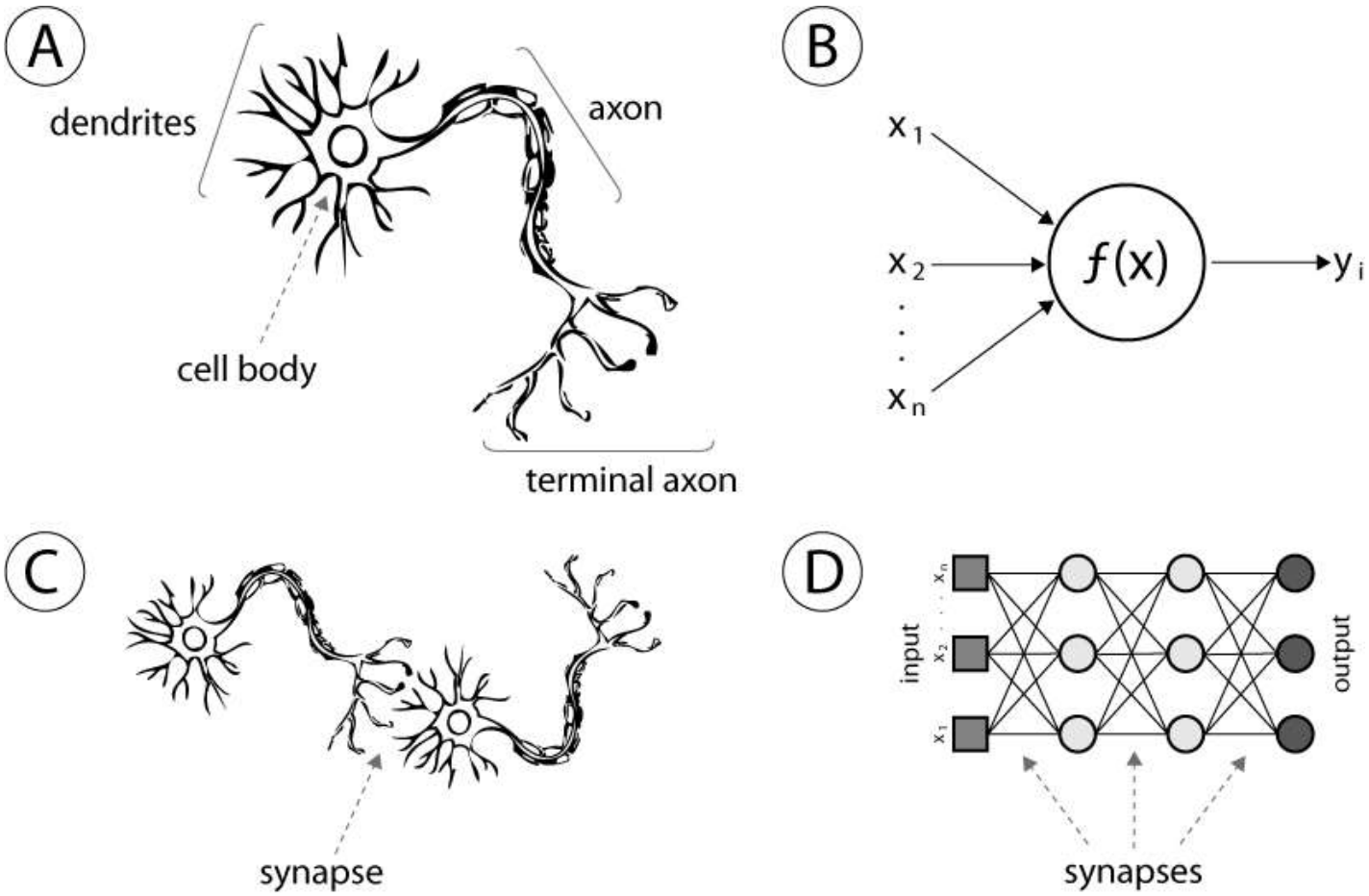
After looking into basic topics, In this assignment we will be starting a new part. From now onwards we will leveling up and studying deep learning topics. Many of the NLP use cases are solved using deep learning algorithms as they are very efficient and fast. So it is very important to first study some of the deep learning topics before diving into NLP use cases. Deep learning is part of a broader family of machine learning methods based on artificial neural networks.




```
###Refer Video
from IPython.display import YouTubeVideo
YouTubeVideo('0aDq6ax6kGQ', width=600, height=300)
```




Artificial neural networks replicate the human biological neurons (nerve cells).




The types of Artificial Neural Networks?

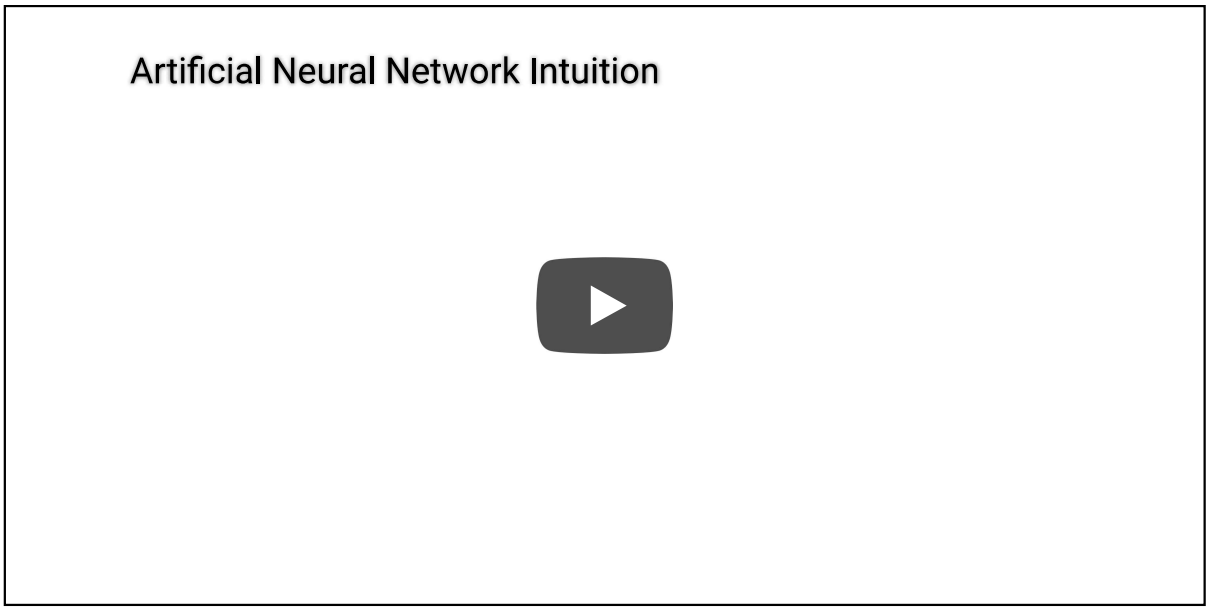
1. Feedforward Neural Network [FNN]
2. Recurrent Neural Network [RNN]
3. Convolutional Neural Network [CNN]

**Artificial Neural Networks [ANN]**:

Artificial Neural Networks contain artificial neurons which are called units. These units are arranged in a series of layers that together constitute the whole Artificial Neural Networks in a system. A

layer can have only a dozen units or millions of units as this depends on the complexity of the system. Commonly, Artificial Neural Network has an input layer, output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

```
#### Refer Video
from IPython.display import YouTubeVideo
YouTubeVideo('O-bCDtHVPtA', width=600, height=300)
```



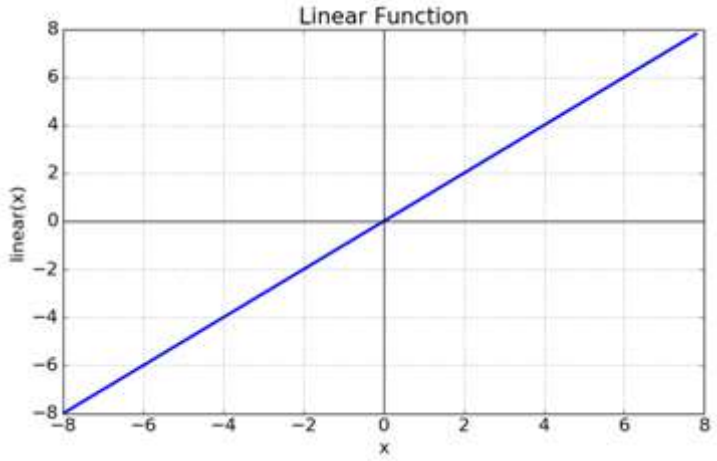

## Types of activation Functions?

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc.

The Activation Functions can be based on 2 types-

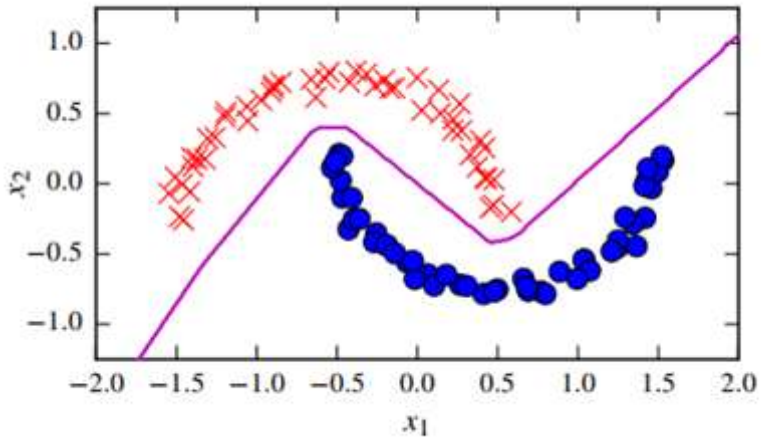
1. Linear Activation Function
2. Non-linear Activation Functions

### Linear Activation Function




As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.
Equation: f(x) = x
Range: (-infinity to infinity)

## Non-linear Activation Function




The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this

It makes it easy for the model to generalize or adapt to a variety of data and to differentiate between the outputs.

The main terminologies needed to understand for nonlinear functions are:
**Derivative or Differential:** Change in y-axis w.r.t. change in the x-axis. It is also known as a slope.
**Monotonic function:** A function that is either entirely non-increasing or non-decreasing.

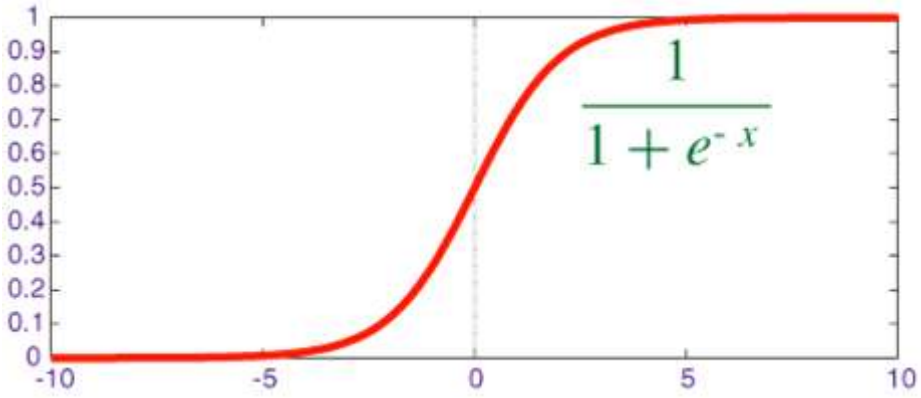
Types of Non-linear Activation Function

1. Sigmoid Activation Function

2. Tanh Activation Function
3. ReLU (Rectified Linear Unit) Activation Function

## 1.Sigmoid Activation Function

The Sigmoid Function curve looks like an S-shape.



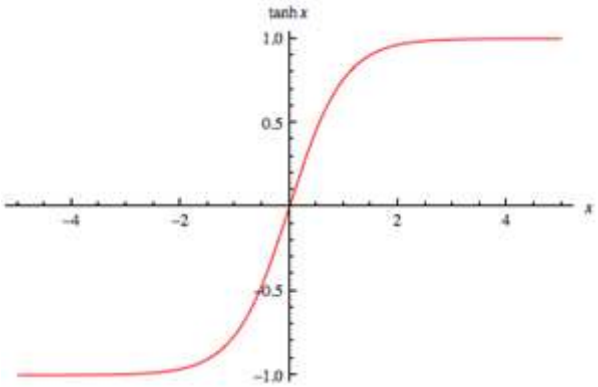

1. We use the sigmoid function is that it exists between (0 to 1).
2. It is especially used for models where we have to predict the probability as an output.
3. The probability of anything exists only between the range of 0 and 1.
4. The function is differentiable,we can find the slope of the sigmoid curve at any two points.
5. The function is monotonic but the function's derivative is not.
6. The softmax function is a more generalized logistic activation function that is used for multiclass classification.

Reference:https://en.wikipedia.org/wiki/Sigmoid_function

```
YouTubeVideo('TPqr8t919YM', width=600, height=300)
```

## 2. Tanh Activation Function

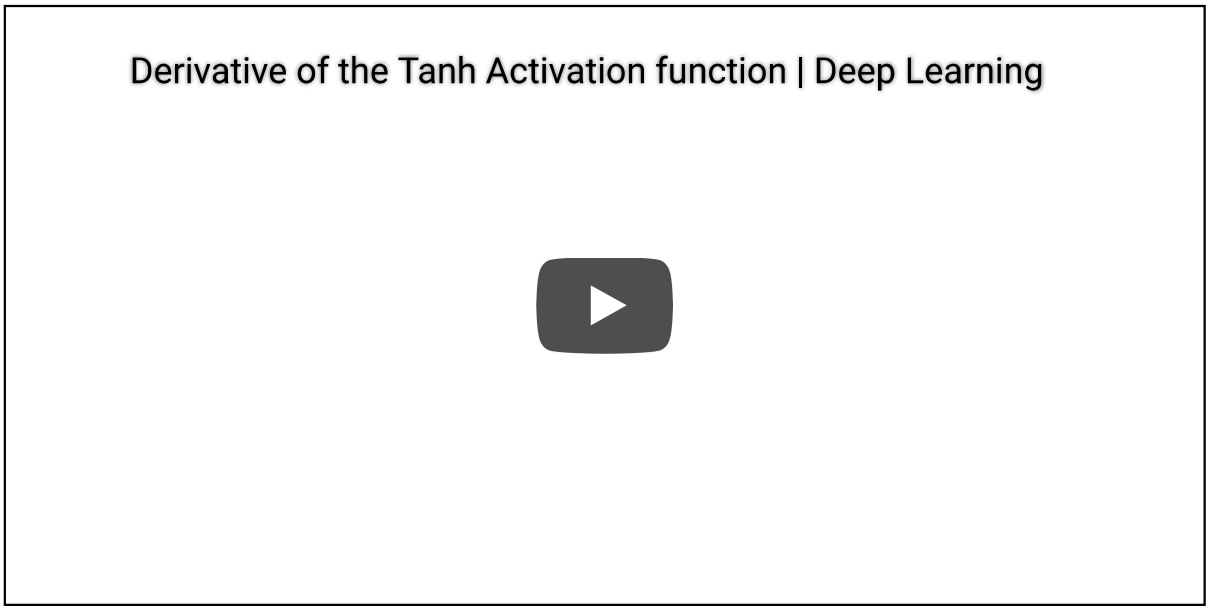
tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s-shaped).




1. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.
2. The function is differentiable.
3. The function is monotonic while its derivative is not monotonic.
4. The tanh function is mainly used classification between two classes.
5. Both tanh and logistic sigmoid activation functions are used in feed-forward nets.

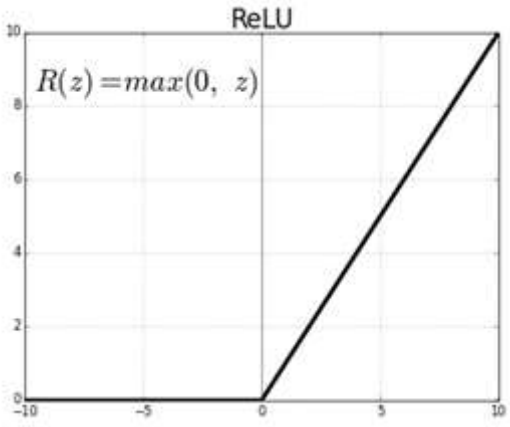
Reference:https://en.wikipedia.org/wiki/Activation_function

```
YouTubeVideo('u0VsKSoSM4Y', width=600, height=300)
```




## 3.ReLU (Rectified Linear Unit) Activation Function

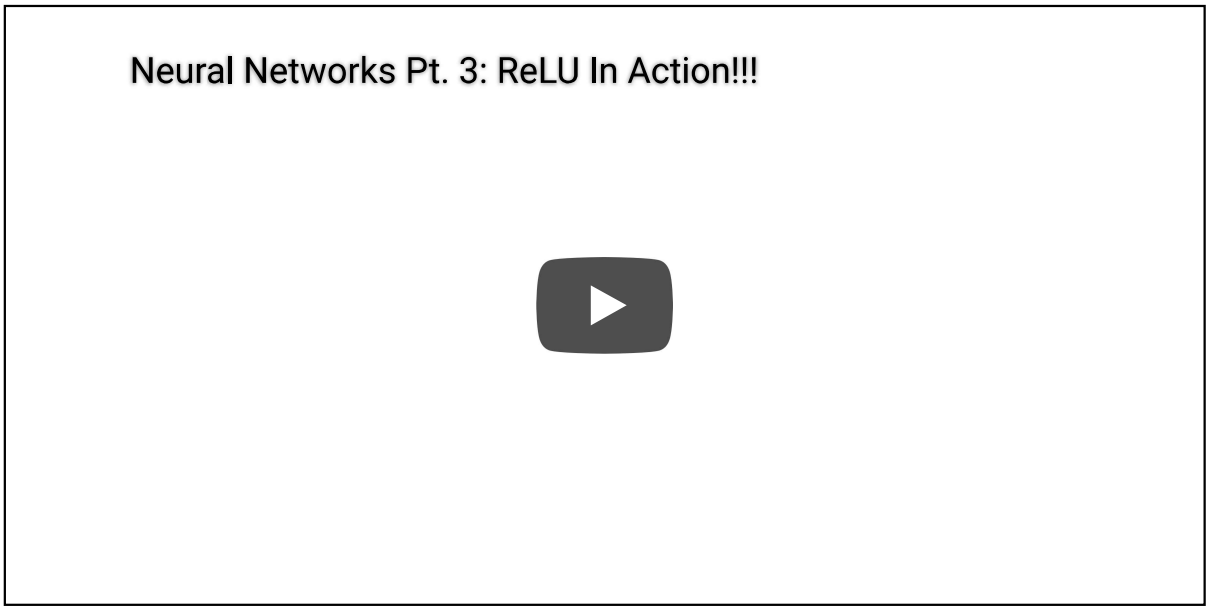
The ReLU is the most used activation function in the world right now. Since it is used in almost all the convolutional neural networks or deep learning.




1. The ReLU is half rectified (from bottom).
2. f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
3. Range: [ 0 to infinity]
4. The function and it is derivative both are monotonic.
5. The issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly.

Reference:https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

```
YouTubeVideo('68BZ5f7P94E', width=600, height=300)
```




## Pratical Implementation of ANN

After looking at the theory part of ANN, let's move ahead with the implementation.

In this ANN Implementation,customer churn is simply the rate at which customers leave doing business with an entity. Simply put, churn prediction involves determining the possibility of customers stopping doing business with an entity. In other words, if a consumer has purchased a subscription to a particular service, we must determine the likelihood that the customer would leave or cancel the membership. It is a critical prediction for many businesses because acquiring new clients often costs more than retaining existing ones. Customer churn measures how and why are customers leaving the business.

You download the dataset from here
https://drive.google.com/file/d/1yPCwRKWMRrIPBmDZtV1rcZJXdaOb0dAU/view?usp=sharing

We will start importing
**Pandas** for data analysis,
**Numpy** for calculating N-dimensional array,
and **Tensorflow** is used for multiple tasks but has a particular focus on the training and inference of deep neural networks and Keras acts as an interface for the TensorFlow library..

```
#import libraries
```

we will define our dataset and then we will see our churn dataset for overview.

```
#loading dataset using pandas
```

```
#partial view of dataset from top
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balan |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.( |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.8 |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.8 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.( |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.8 |

```
#dimension of dataset using shape
```

```
(10000, 14)
```

In this dataset there are 100000 rows and 14 columns are present. There are some categorical and some numerical columns present.

Now time to preprocess the data,
firstly we will observe the dataset, this means we have to see the data types of the columns. we will check the dataset information using the info().

```
#basic dataset information
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

You can see that the datatypes of each column, number of rows present with non-null values, there are many int float, and remaining are string datatype columns.

Now we will summarize the statistical part by usning describe method.

```
#basic statistics symmary
```

| | count | mean | std | min | 25% | 5( |
|---|---|---|---|---|---|---|
| **RowNumber** | 10000.0 | 5.000500e+03 | 2886.895680 | 1.00 | 2500.75 | 5.000500e+( |
| **CustomerId** | 10000.0 | 1.569094e+07 | 71936.186123 | 15565701.00 | 15628528.25 | 1.569074e+( |
| **CreditScore** | 10000.0 | 6.505288e+02 | 96.653299 | 350.00 | 584.00 | 6.520000e+( |
| **Age** | 10000.0 | 3.892180e+01 | 10.487806 | 18.00 | 32.00 | 3.700000e+( |
| **Tenure** | 10000.0 | 5.012800e+00 | 2.892174 | 0.00 | 3.00 | 5.000000e+( |
| **Balance** | 10000.0 | 7.648589e+04 | 62397.405202 | 0.00 | 0.00 | 9.719854e+( |
| **NumOfProducts** | 10000.0 | 1.530200e+00 | 0.581654 | 1.00 | 1.00 | 1.000000e+( |
| **HasCrCard** | 10000.0 | 7.055000e-01 | 0.455840 | 0.00 | 0.00 | 1.000000e+( |
| **IsActiveMember** | 10000.0 | 5.151000e-01 | 0.499797 | 0.00 | 0.00 | 1.000000e+( |

Now we have to check for null values, for this, we use the pandas IsNull() method which will give True if the null value is present and False when there are no null values.

```
#checking for missing values
print(str('Any missing data or NaN in the dataset:'),_____your code here______
```

```
Any missing data or NaN in the dataset: RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

As we can see there is no Nan values in dataset but there some categorical features.

As there is no importance in cust id, row no and sur name for modelling we are not included here in independent feature

```
#assigning independant features without including the cust id,row no and surname.

#assigning dependant feature i.e churn
```

```
#independent features
```

```
[[619 'France' 'Female' ... 1 1 101348.88]
 [608 'Spain' 'Female' ... 0 1 112542.58]
 [502 'France' 'Female' ... 1 0 113931.57]
 ...
 [709 'France' 'Female' ... 0 1 42085.58]
 [772 'Germany' 'Male' ... 1 0 92888.52]
 [792 'France' 'Female' ... 1 0 38190.78]]
```

```
#dependent features
```

```
[1 0 1 ... 1 1 0]
```

As we have two columns with categorical values we go for encoding, we need to convert them into numerical values i.e Categorical encoding
Gender will have some correlation with other features so it is necessary to keep that feature in the data so we go for label encoding. For that we import the LabelEncoder library from sklearn.preprocessing. Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form.
Refer documentation for label encoding: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/.
Next we create labelEncoder object for using features of converting into machine readable form. After creating object of labelEncoder We will fit and transform the gender column into it for machine readable form.

```
#importing LabelEnoder from sklearn

#gender column in index 2
#X[:, 2] = le.fit_transform(X[:, 2])
#The above line is given for fit and transform of gender column
```

The feature country has more than two unique values, so OneHotEncoding has to be used for encoding the column. Import the libraries ColumnTransformer and OneHotEncoder. ColumnTransformer allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space.
Refer documentation: https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html
In OneHotEncoding, each category value is converted into a new column and assigned a 1 or 0

(notation for true/false) value to the column.

Refer documentation: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.

The ct object of the ColumnTransformer class is created using the parameters transformers and remainder. transformers parameter contains the encoder name and the column index to be encoded. As country name is the second column, index is set to 1. remainder parameter has value that decides whether to include only the specified column or other columns as well in the output. 'passthrough' value is passed to remainder which ensures that only the specified column(country name) is transformed and combined in the output.

```
###Refer video
YouTubeVideo('ZS0hzcA5w9I', width=600, height=300)
```


```
# Importing ColumnTransformer,OneHotEncoder

#country name is present in 1st index value


#printing X Values.
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]
 [1.0 0.0 0.0 ... 1 0 38190.78]]
```

we have to split our dataset into train and test sets, where the training set is used to train the model, and the testing set is used for testing the values of targeted columns.

```
#training and testing split
```

We have just imported the train_test_split() method from the sklearn and we set some parameters where testing size was 30% and the remaining 70% considered as training data.

StandardScaler : It transforms the data in such a manner that it has mean as 0 and standard deviation as 1. In short, it standardizes the data. Standardization is useful for data which has negative values. It arranges the data in a standard normal distribution. It is more useful in classification than regression.
Refer for Documentation: [https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html)

```
#feature scaling is an important and mandatory for ann process before modelling for X train a
#importing StandScaler from sklearn

#creating Standscalar

#fit and transform of X train and X test
```

we have to define our model, which means we have to set the parameters and layers of the deep neural network which will be used for training the data.

```
#ANN - initializing
# Initializing the ANN by calling the Sequential class from keras of Tensorflow
```

we define sequential model, in the sequential model the input, hidden and output layers are connected into the sequential manner, here we define one input layer which contains all 6 columns as an input, second is hidden layers which contain 6,here we apply RelU activation function. Our last layer is the output layer, as our output is in the form of 1 and 0 so, we will use the sigmoid activation function.

```
#input layer
# 6 features
```

```
# Adding "fully connected" INPUT layer to the Sequential ANN by calling Dense class
# Number of Units = 6 and Activation Function = Rectifier
```

Here we have added a input layer in ann variable with 6 units and applying RelU activation function by using dense function.

```
#hidden layer
# Adding "fully connected" SECOND layer to the Sequential ANN by calling Dense class
# Number of Units = 6 and Activation Function = Rectifier
```

As we have added a hiddien with same parameter because it gives better accuary using this parameter.
You try with our units for playing with hidden layers or by adding multiple hidden layers :)

```
#output layer
#as target value is binary - AF
# Adding "fully connected" OUTPUT layer to the Sequential ANN by calling Dense class
# we use sigmoid for binary output
# Number of Units = 1 and Activation Function = Sigmoid
```

we compile our sequential model and fit the training data into our model. The compilation of the model is the final step of creating an artificial neural model. The compile defines the loss function, the optimizer, and the metrics which we have to give into parameters.

Here we use compile method for compiling the model, we set some parameters into the compile method.

```
#loss - target is binary
# Compiling the ANN
# Type of Optimizer = Adam Optimizer, Loss Function =  crossentropy for binary dependent vari
```

now we fit our model to training data for 50 epochs and batch size=32.by using the X train and y train.

```
#training set
```

```
     Epoch 1/50
```

```
219/219 [==============================] - 1s 2ms/step - loss: 0.5425 - accuracy: 0.7
Epoch 2/50
219/219 [==============================] - 0s 1ms/step - loss: 0.4544 - accuracy: 0.8
Epoch 3/50
219/219 [==============================] - 0s 1ms/step - loss: 0.4321 - accuracy: 0.8
Epoch 4/50
219/219 [==============================] - 0s 1ms/step - loss: 0.4194 - accuracy: 0.8
Epoch 5/50
219/219 [==============================] - 0s 1ms/step - loss: 0.4078 - accuracy: 0.8
Epoch 6/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3947 - accuracy: 0.8
Epoch 7/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3802 - accuracy: 0.8
Epoch 8/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3678 - accuracy: 0.8
Epoch 9/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3595 - accuracy: 0.8
Epoch 10/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3548 - accuracy: 0.8
Epoch 11/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3512 - accuracy: 0.8
Epoch 12/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3491 - accuracy: 0.8
Epoch 13/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3466 - accuracy: 0.8
Epoch 14/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3460 - accuracy: 0.8
Epoch 15/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3445 - accuracy: 0.8
Epoch 16/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3437 - accuracy: 0.8
Epoch 17/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3431 - accuracy: 0.8
Epoch 18/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3420 - accuracy: 0.8
Epoch 19/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3416 - accuracy: 0.8
Epoch 20/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3412 - accuracy: 0.8
Epoch 21/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3402 - accuracy: 0.8
Epoch 22/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3401 - accuracy: 0.8
Epoch 23/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3392 - accuracy: 0.8
Epoch 24/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3390 - accuracy: 0.8
Epoch 25/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3389 - accuracy: 0.8
Epoch 26/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3385 - accuracy: 0.8
Epoch 27/50
219/219 [==============================] - 0s 1ms/step - loss: 0.3381 - accuracy: 0.8
Epoch 28/50
219/219 [==============================] - 0s 2ms/step - loss: 0.3374 - accuracy: 0.8
Epoch 29/50
```

## Now try with your epochs and batch size for better understanding -__-

we are ready for predication beacuse we are ready with our training model. We will predict using X_test from trainning model.

```
#test result - prediction
```

```
[[0 0]
 [0 1]
 [0 0]
 ...
 [0 0]
 [0 0]
 [1 1]]
```

Performance Matrices this is used in the classification problems, and the customer churn is also a classification problem so we use performance metrics for checking the model behavior.

At the last, we have to predict the churn which is in the form of 0 and 1 means it was a classification problem, and the performance of the classification problem is observed with the performance metrics.

What is confusion matrix in Sklearn?
A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier

```
###Refer Video
YouTubeVideo('Kdsp6soqA7o', width=600, height=300)
```

## Machine Learning Fundamentals: The Confusion Matrix

```
#import confusion matix and accuracy score from sklearn and find thier matrix and accuracy
#accuracy and confusion matrix
```

```
[[2253  126]
 [ 296  325]]
0.8593333333333333
```

You can also try with anothers set of inputs by changing the test size or adding multiple hidden layers or changing the units values for better accuracies:))

That's cool, 85% is the accuracy of our model.

We have come to an end of this project but don't stop here, try as many projects of the similar type to get a better understanding of the use cases. Solve the practice sheet of this project to test yourself.!!

Great job!! You have come to the end of this assignment. Treat yourself for this :))

# Do fill this feedback form

You may head on to the next assignment.