

Table of Content

S No	Contents	Page No
I	Introduction	2
II	History of Artificial Neural Network <ul style="list-style-type: none"> • MULTI-LAYER PERCEPTRONS • CONVOLUTIONAL NEURAL NETWORKS • RECURRENT NEURAL NETWORKS 	4
III	Statistical Learning <ul style="list-style-type: none"> • supervised statistical learning • unsupervised statistical learning 	10
IV	Methodology <ul style="list-style-type: none"> • Artificial Neural Network • The architecture of an Artificial Neural Network • Forward Propagation= weight initialization, Activation Function • Backward Propagation Loss Functions, Optimizer • Evaluation of Measures=Accuracy, Precision, Recall, F-measure, Sensitivity, Specificity 	16
V	Data Description <ul style="list-style-type: none"> • Dataset Description, • Exploratory Data Analysis • Outlier =Z-Score, Mahalanobis distance • Feature Engineering • Feature Scaling • Model Selection 	39
VI	Data Analysis <ul style="list-style-type: none"> • Data Understanding • Visualizing the data • Outlier Detection • Model Result 	64
V11	Conclusion	71

Introduction

There are many competitors in the banking sector nowadays and those competitors are also ready to provide higher quality and lower prices for the same products and services. So, customers are shifting loyalties from bank to bank. When products or services do occur, customer churn or attrition occurs. *There are two kinds of churns like voluntary or involuntary churning.*

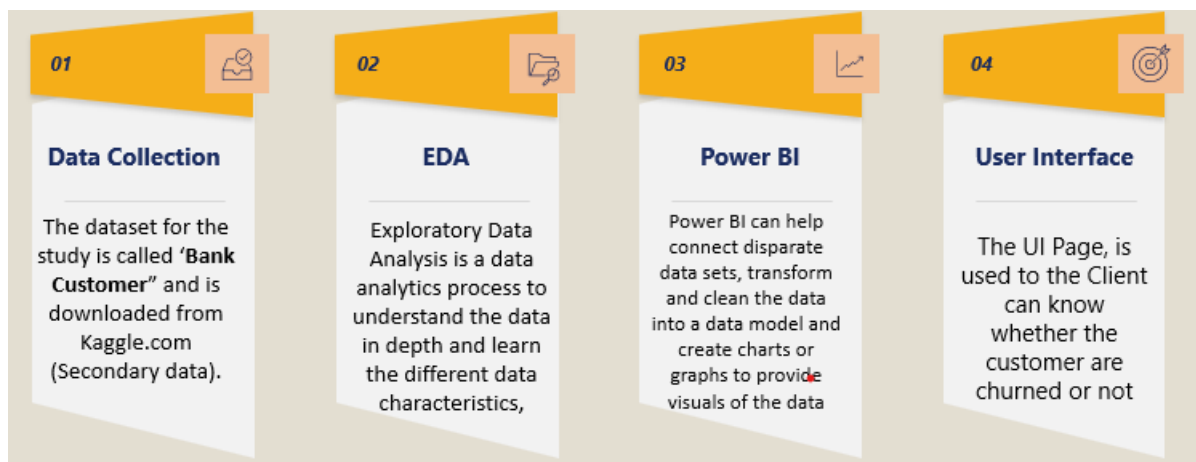
As a result of increasing competition, it is important for banks to maintain existing customers, as this is more cost-effective than acquiring new ones, in order to ensure their position in society. The customer leaves the bank or stops using Involuntary churners which the bank removes users. Voluntary churn arose when customers immediately stopped using products or services. This study is focused on the voluntary churn. *To prevent this, the prediction of customer churn is necessary to determine whether customers can discontinue products or services.* Taken together, there is a growing demand for customer churn analysis which studies a set of characteristics in order to predict customer churn

This has intensified the demand for predictive modelling built on for example statistical learning methods. Churn prediction is mainly regard as a sort of binary classification problem. However, churn studies can focus on churn rate, and this makes them no longer a binary classification This churn prediction is based on socio-demographic attributes, balance, account level and behavioural attributes of the customers. Models were developed using data mining techniques to forecast possible churns with satisfactory performance. Customer's transaction patterns and other factors directly or indirectly affecting the customer's choices are mainly used for the prediction. This low-churn rate resulting from imbalanced nature makes it difficult to achieve an actual correct classification rate by traditional machine learning methods. Because of that, machine learning-based classification gives high accuracy rates but rather low precision in terms of predicting the churning's class as a performance indicator

These churn prediction confusions demonstrate that a margin of performance improvement is still needed in this area, keeping possibilities emerging in big data analytics. Traditional machine learning methods contain two main drawbacks. First, they usually require so much time to extract features from thousands of customer attributes, Besides, an expert is mainly needed. Second, traditional churn models are usually developed and applied for a special dataset. *Deep learning (DL) methods* could be utilized in overcoming these limitations of traditional churn models.

First of all, they are faster than traditional methods. Moreover, DL approaches can determine beneficial features without any prior knowledge and expert support. Therefore, [applying a DL method for churn prediction in the Bank Customer churn prediction](#) , including many data attributes, can produce high accuracy rates in the prediction results. The experimental results showed that the ANN based prediction method produces better accuracy rates than traditional models without manual feature extraction.

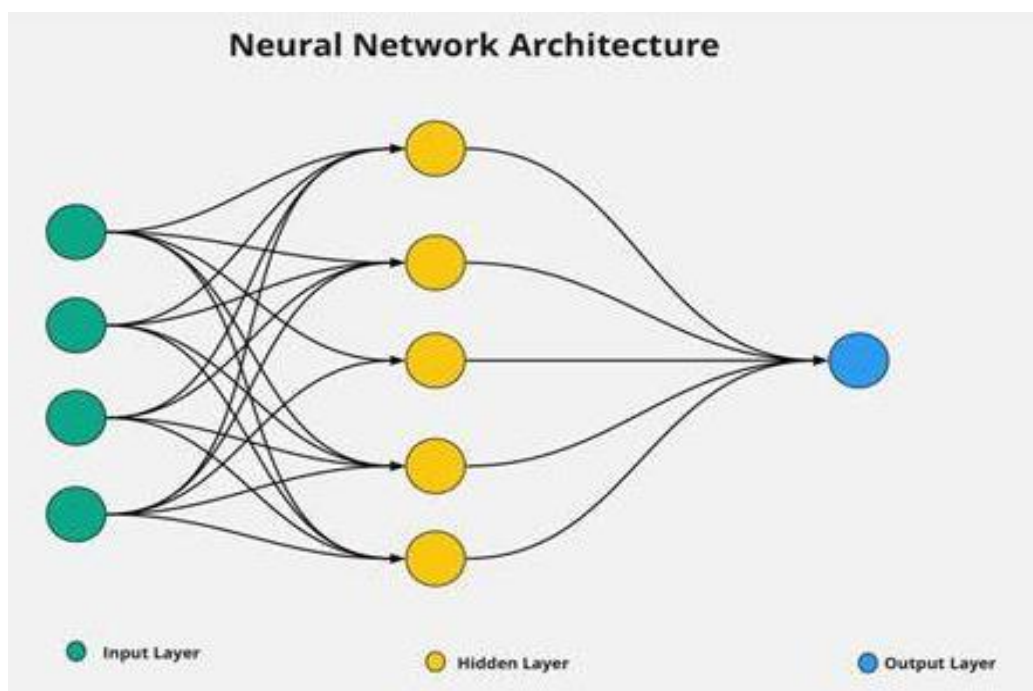
Objective



HISTORY OF ARTIFICIAL NEURAL NETWORK :

The history of **neural networking** arguably began in the late **1800s** with scientific endeavors to study the activity of the human brain. In **1890**, **William James** published the first work about brain activity patterns. In **1943**, **McCulloch** and **Pitts** created a model of the neuron that is still used today in an **artificial neural network**. This model is segmented in two parts.

- A summation over-weighted inputs.
- An output function of the sum.



How do neural networks work?

Think of each individual node as its own [linear regression](#) model, composed of input data, weights, a bias (or threshold), and an output. The formula would look something like this:

$$\sum w_i x_i + \text{bias} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{bias}$$

$$\text{output} = f(x) = 1 \text{ if } \sum w_i x_i + b \geq 0; 0 \text{ if } \sum w_i x_i + b < 0$$

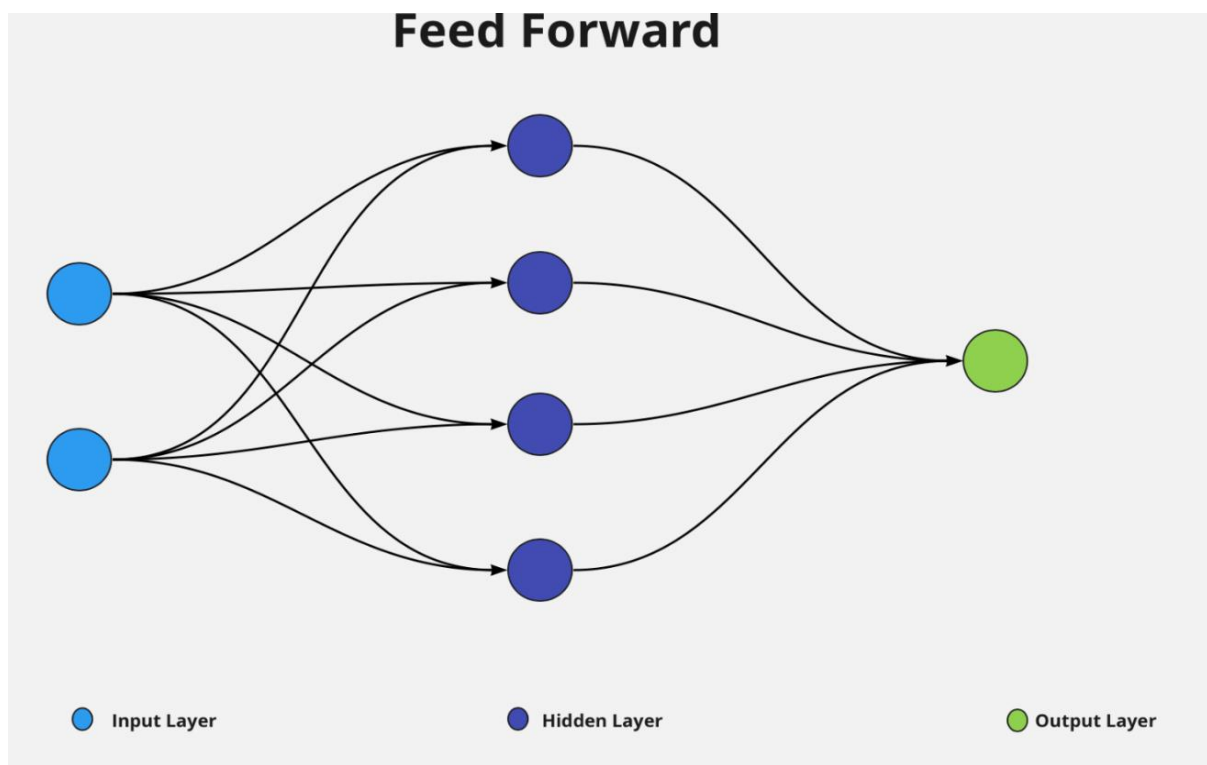
Types of neural networks:

Neural networks can be classified into different types are

1. Feedforward neural networks, or multi-layer perceptrons (MLPs)
2. Convolutional neural networks (CNNs)
3. Recurrent neural networks (RNNs)

FEEDFORWARD NEURAL NETWORKS, OR MULTI-LAYER PERCEPTRONS (MLPS):

They are comprised of an input layer, a hidden layer or layers, and an output layer. While these neural networks are also commonly referred to as MLPs, it's important to note that they are actually comprised of sigmoid neurons, not perceptrons, as most real-world problems are nonlinear. Data usually is fed into these models to train them, and they are the foundation for computer vision, [natural language processing](#), and other neural networks.

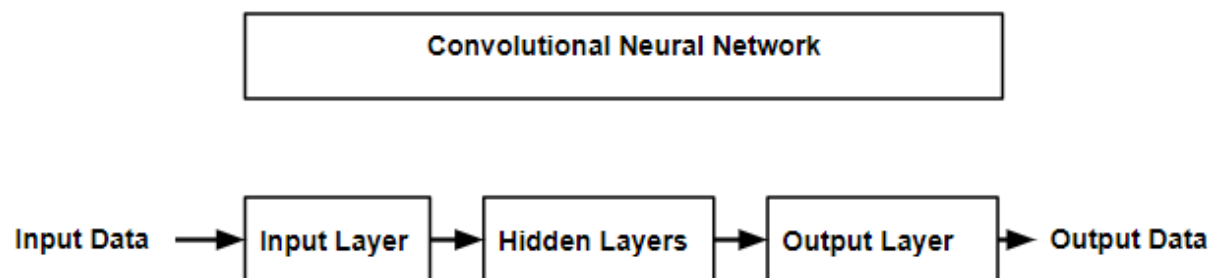


Applications of Feed Forward Neural Networks:

While Feed Forward Neural Networks are fairly straightforward, their simplified architecture can be used as an advantage in particular machine learning applications. For example, one may set up a series of feed forward neural networks with the intention of running them independently from each other, but with a mild intermediary for moderation. Like the human brain, [this process relies on many individual neurons in order to handle and process larger tasks](#). As the individual networks perform their tasks independently, the results can be combined at the end to produce a synthesized, and cohesive output.

CONVOLUTIONAL NEURAL NETWORKS (CNNs):

similar to feedforward networks, but they're usually utilized for image recognition, pattern recognition, and/or computer vision. These networks harness [principles from linear algebra, particularly matrix multiplication, to identify patterns within an image](#).



How Does Convolutional Neural Network work?

Convolutional Neural Network structure consists of four layers:

Convolutional layer:

The convolutional layer is where the action begins. The convolutional layer is designed to discover image features. Usually, it progresses from the general (i.e., shapes) to specific (i.e., identifying elements of an object, recognizing the face of a certain man, etc.).

Rectified Linear Unit layer (aka ReLu):

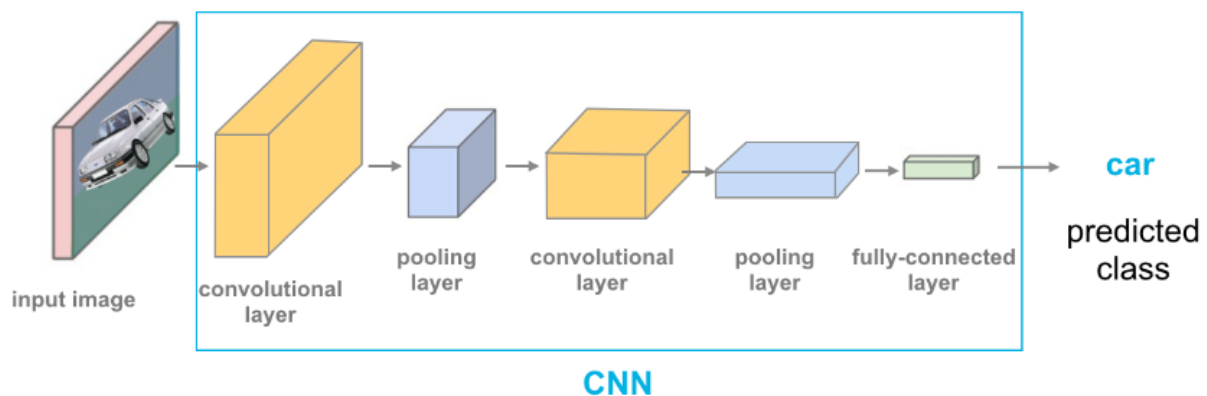
This layer is considered as an extension of a convolutional layer. The goal of ReLu is to increase the image's non-linearity. It is the technique of removing excess fat from a picture in order to improve feature extraction.

Pooling layer:

The pooling layer is used to minimize the number of input parameters, i.e., to conduct regression. In other words, it focuses on the most important aspects of the information obtained.

Connected layer:

It is a standard feed-forward neural network. It's the last straight line before the finish line, where everything is already visible. It's only a matter of time until the results are confirmed.

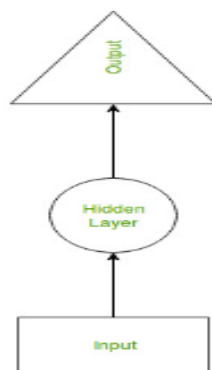


Applications of Convolutional Neural Networks

1. Facial Recognition
2. Medical Image Computing
3. Autonomous Driving
4. Document Analysis

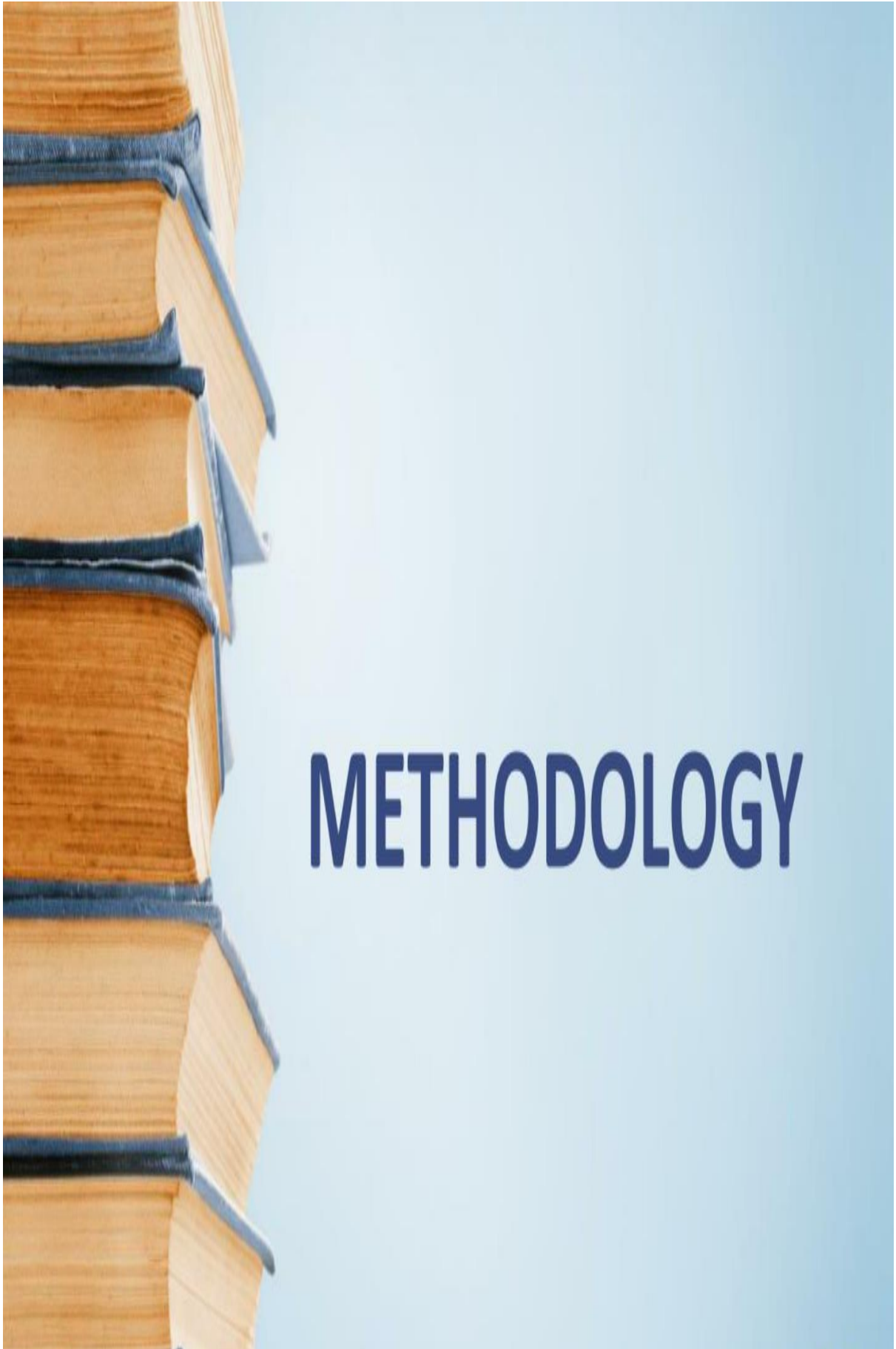
RECURRENT NEURAL NETWORKS (RNNS):

Recurrent Neural Network(RNN) is a type of Neural Network where the **output from the previous step are fed as input to the current step**. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is **Hidden state**, which remembers some information about a sequence.



APPLICATION OF RECURRENT NEURAL NETWORK:

1. Machine Translation
2. Speech Recognition
3. Text Summarization
4. Generating Image Descriptions
5. Generating Text



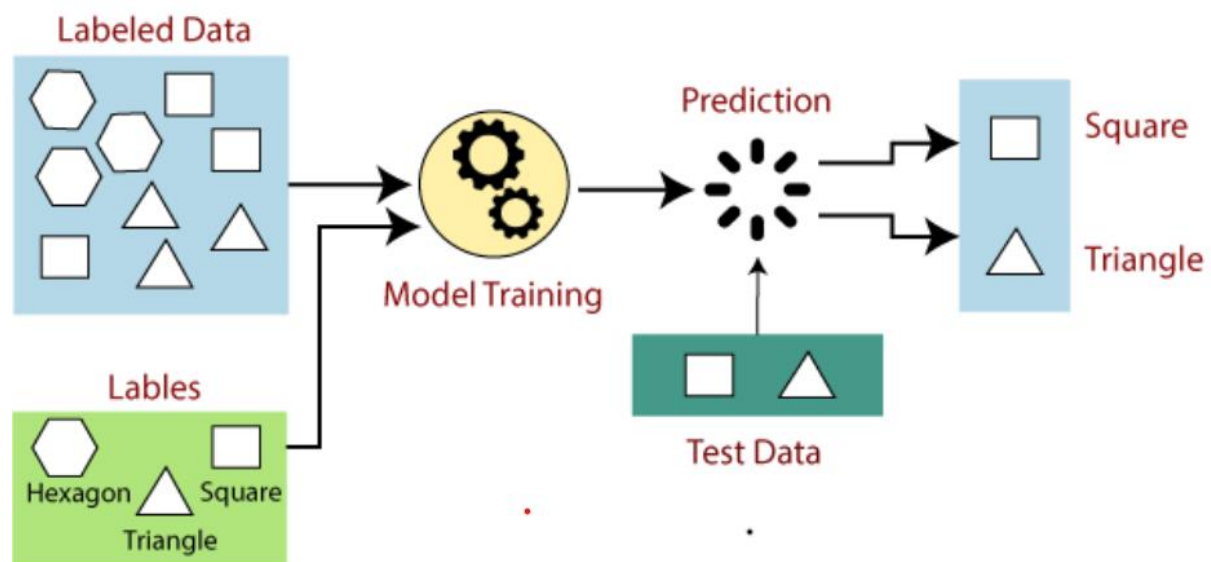
Theoretical Background: [Introduction to Statistical Learning](#).

Statistical learning is a tool to understand data. Most problems fall into one of two approaches, supervised or unsupervised statistical learning.

[Supervised Learning](#):

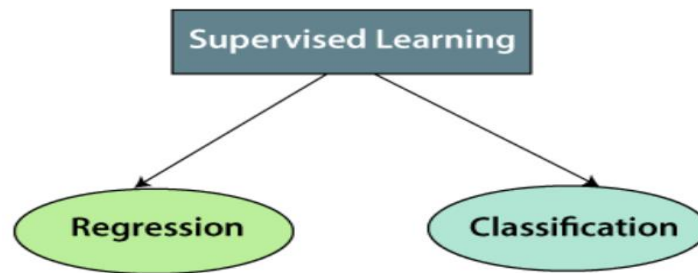
For each observation of the predictor measurement(s) x_i , $i = 1, \dots, n$ has an associated dependent variable y_i . The objective is to fit a model that accurately predicts a response for an unseen observation. Linear and logistic regression are two examples of supervised statistical learning methods

[The working of Supervised learning](#)



Supervised statistical learning aims to predict an outcome (predictive) or understand a relationship between variables (inference). The goal determines what method is preferred. In general, restrictive models are used for [Regression](#) (Inferential statistical learning) and [classification](#) models are used for predictive statistical learning.

Types of supervised Machine learning Algorithms:



Regression Methods:

For predictive statistical learning the goal is to predict an outcome Y . A relationship between x_i and Y is assumed and can generally be written as,

$$Y = f(x) + \epsilon$$

where, ϵ is an error term and has mean 0 and is independent from x_i

Practically, in order to predict Y one needs to estimate the functional form of $f(x)$. This is done by estimating $f(x)$ using the systematic information given by the independent variables x_i . This can be expressed as,

$$\hat{Y} = \hat{f}(x)$$

Moreover, accuracy of the prediction depends on two things. Firstly, the reducible error which can be reduced using the appropriate statistical learning technique. Secondly, the irreducible error which is caused by chance and can thus not be reduced. This is because Y is a function of the error term ϵ which by definition cannot be predicted.

Commonly used Algorithms are :

1. Linear Regression
2. Decision Tree
3. Support Vector Regression
4. Lasso Regression
5. Random Forest

Classification Methods:

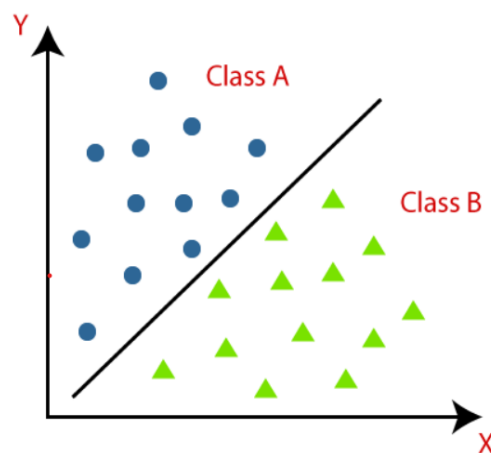
The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups.

For Example , **Yes or No, 0 or 1, Spam or Not Spam, cat or dog**, etc. Classes can be called as targets/labels or categories. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

$$y=f(x)$$

where **y = categorical output**

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.



There are two types of Classifications:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier.
Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.
Example: Classifications of types of crops, Classification of types of music.

In the classification problems, there are two types of learners:

- **Lazy Learners:**

Lazy Learner firstly stores the training dataset and wait until it receives the test dataset. In Lazy learner case, classification is done on the basis of the most related data stored in the training dataset. It takes less time in training but more time for predictions.

Example:

K-NN algorithm, Case-based reasoning

- **Eager Learners:**

Eager Learners develop a classification model based on a training dataset before receiving a test dataset. Opposite to Lazy learners, Eager Learner takes more time in learning, and less time in prediction.

Example:

Decision Trees, Naïve Bayes, ANN.

Types of ML Classification Algorithms:

Classification Algorithms can be further divided into the Mainly two category:

- **Linear Models**

1. Logistic Regression
2. Support Vector Machines

- **Non-linear Model**

1. K-Nearest Neighbours
2. Kernel SVM
3. Naïve Bayes
4. Decision Tree Classification
5. Random Forest Classification
6. **ARTIFICIAL NEURAL NETWORK (ANN)**

The models are trained using labeled data under the supervision of training data. But there may be many cases in which **we do not have labeled data** and **need to find the hidden patterns from the given dataset**. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

Unsupervised Learning

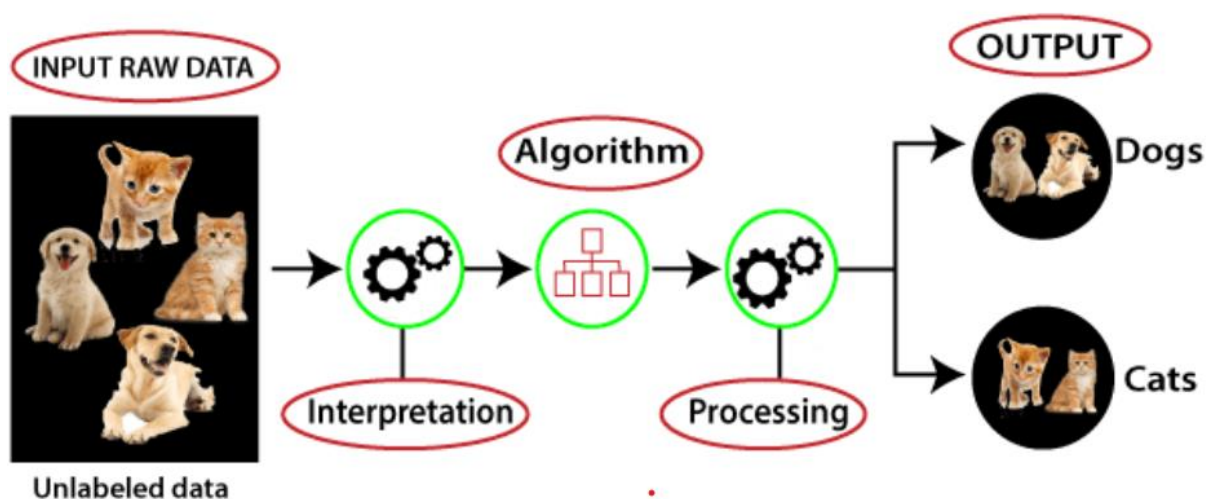
unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.**

Importance of Unsupervised Learning:

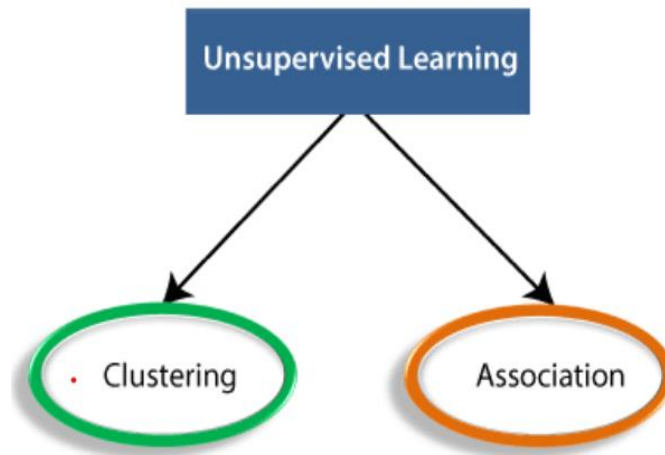
- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning



Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering:**

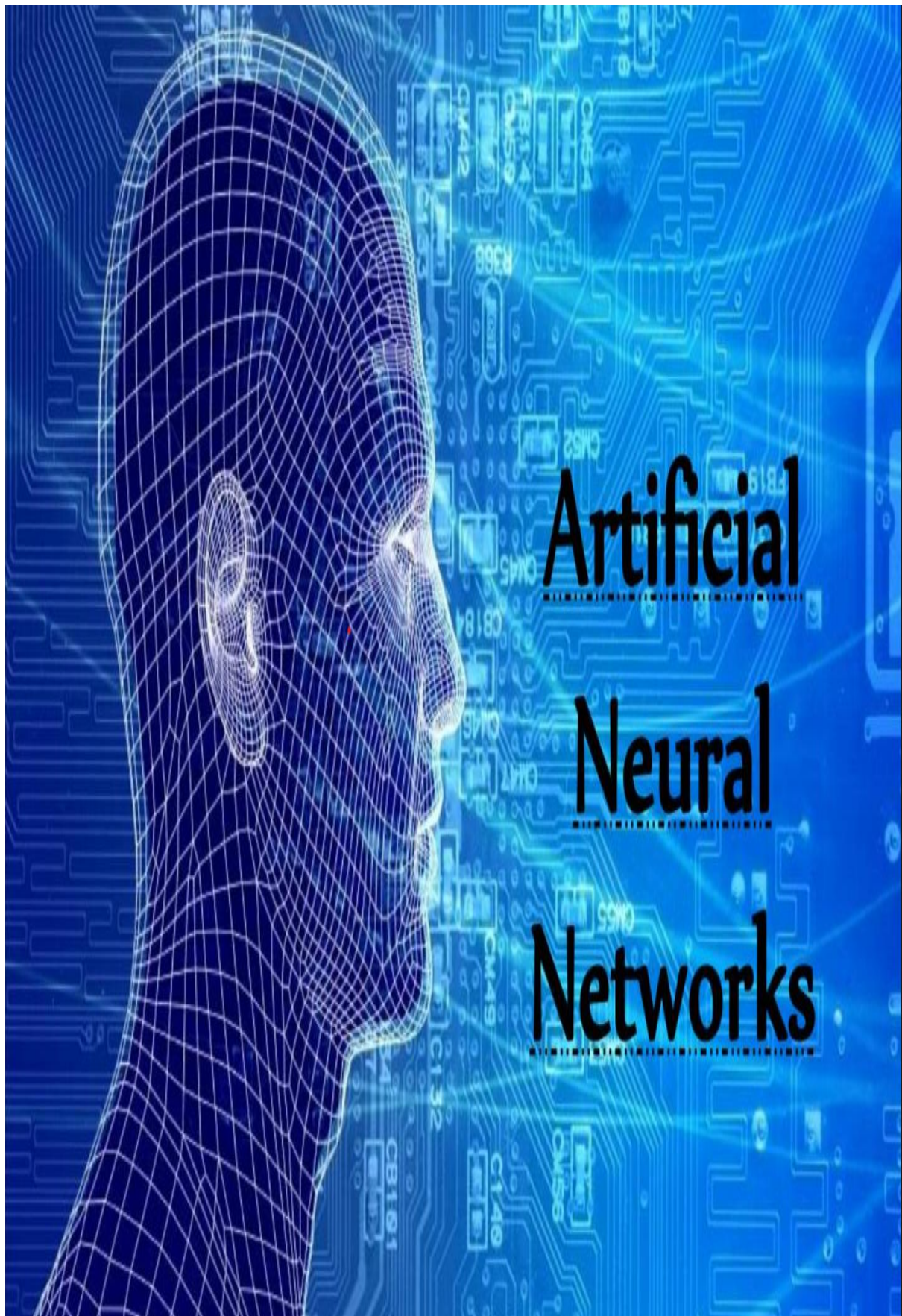
Clustering is a method of **grouping the objects into clusters** such that objects with most similarities remain in a group and have less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

- **Association:**

An association rule is an unsupervised learning method which is used for **finding the relationships between variables in the large database**. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Commonly used UnSupervised Algorithms

1. K-means clustering
2. KNN (k-nearest neighbors)
3. Hierarchical clustering.
4. Anomaly detection.
5. Independent Component Analysis.
6. Factor Analysis



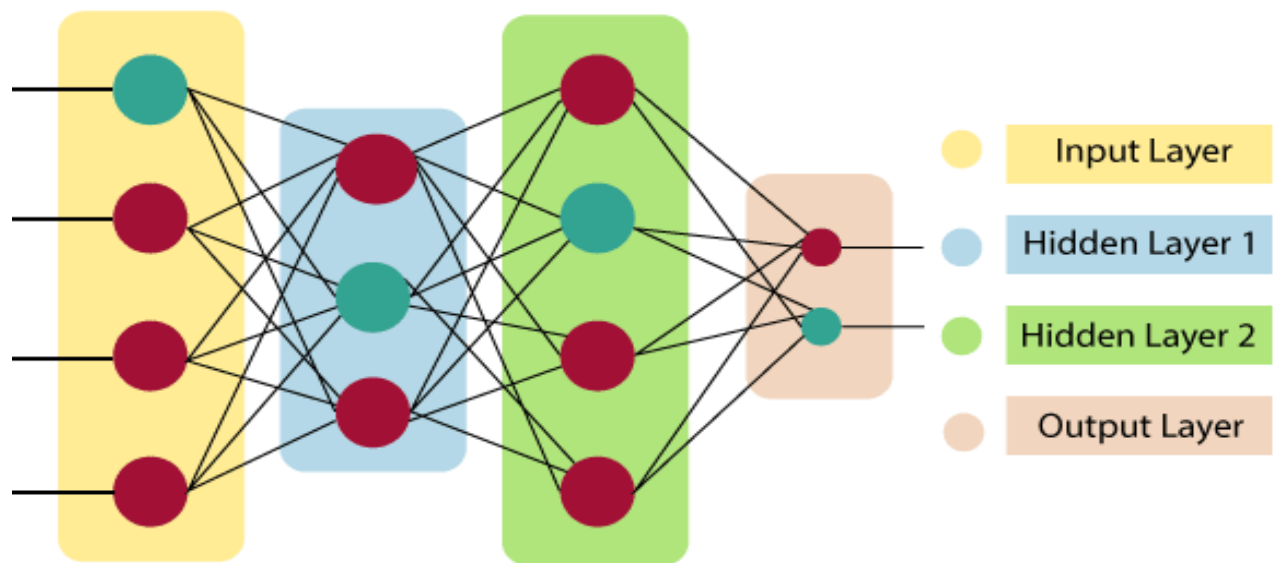
ARTIFICIAL NEURAL NETWORK (ANN):

- In 1949, Donald Hebb published "**The Organization of Behaviour**," which illustrated a law for synaptic neuron learning. This law, later known as **Hebbian Learning** in honour of Donald Hebb, is one of the most straightforward and simple learning rules for artificial neural networks.
- In 1951, **Narvin Minsky** made the **first Artificial Neural Network (ANN)** while working at Princeton.
- In 1958, "**The Computer and the Brain**" were published, a year after **Jhon von Neumann's** death. In that book, von Neumann proposed numerous extreme changes to how analysts had been modeling the brain.

Artificial Neural Network Model

- The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.
- Artificial neural networks (ANNs) or simulated neural networks (SNNs), are a **subset of machine learning and are at the heart of deep learning algorithms**
- Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.
- ANN is an information processing model inspired by the biological neuron system. It is composed of a **large number of highly interconnected processing elements** known as the neuron to solve problems.

The architecture of an Artificial Neural Network :

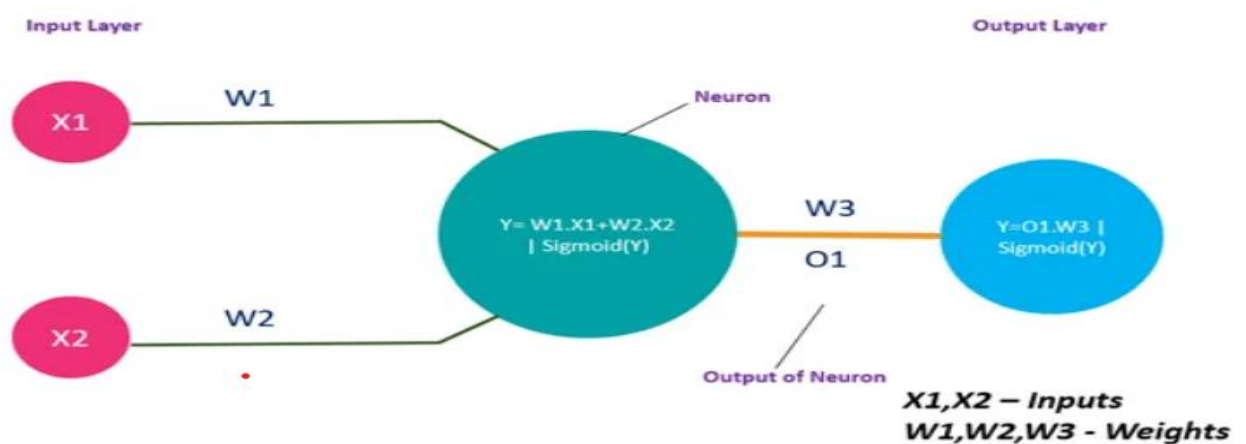


Working of Neural Network

A neural network works based on two principles

1. **Forward Propagation**
2. **Backward Propagation**

Forward Propagation



1. Considering the data and would like to apply binary classification (yes or No) to get the desired output.
2. Take a sample having features as X1, X2, and these features will be operated over a set of processes to predict the outcome.
3. Each feature is associated with a weight, where X1, X2 as features and W1, W2 as weights. These are served as input to a neuron.
4. A neuron performs both functions.

a) Summation

b) Activation.

The inputs given to a perceptron (A simple artificial neuron having an input layer and output layer is called a perceptron) are processed by Summation function and followed by activation function to get the desired output.

5. Summation function

All features are multiplied by their weights and bias are summed up.

$$Y = W_1X_1 + W_2X_2 + \dots + W_nX_n + b.$$

Where

X1 = are the independent variables or the inputs

W1 = Weight of the corresponding input variable

B = Bias or intercept

6. This summed function is applied over an Activation function. The output from this neuron is multiplied with the weight W3 and supplied as input to the output layer.
7. The same process happens in each neuron, but we vary the activation functions in hidden layer neurons, not in the output layer.

The default weight initialization in keras = Glorot Uniform

Glorot Uniform Initialization

Glorot Uniform initialization is **the method which generate the initial value from Uniform distribution**. Each layer has different max and min value of Uniform distribution depends on the number of input units and output units.

Glorot Uniform Distribution

In Glorot Uniform Distribution , weights belong to uniform distribution in range of a and b defined as below:

$$W \approx U(a,b)$$

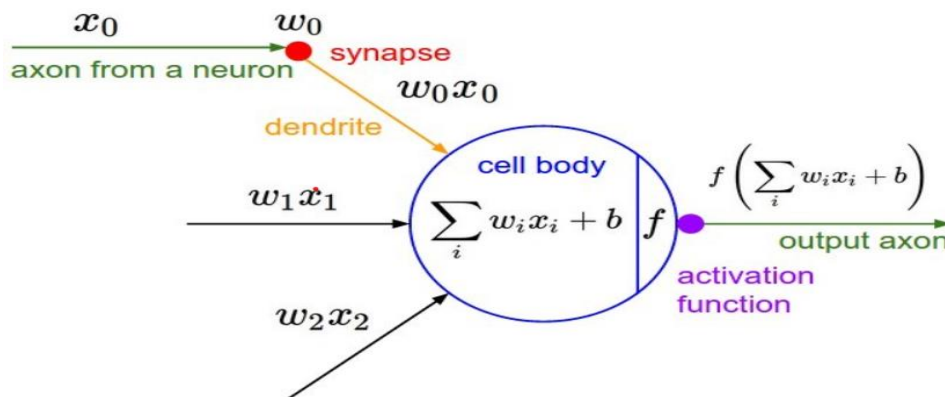
$$a = -\sqrt{\frac{6}{f_{in} + f_{out}}}, \quad b = \sqrt{\frac{6}{f_{in} + f_{out}}}$$

- The term kernel_initializer is a fancy term for which statistical distribution or function to use for initialising the weights. In case of statistical distribution, the library will generate numbers from that statistical distribution and use as starting weights
- Weight initialization is used to define the initial values for the parameters in neural network models prior to training the models on a dataset
- Glorot Initialization is used to maintain the same smooth distribution for both the forward pass as well the backpropagation.

Activation Function

- Activation functions are mathematical equations that helps to determine the output of a neural network. These type of function are attached to each neuron in the network, and determines whether it should be activated or not, based on whether each neuron's input is relevant for the model's prediction.

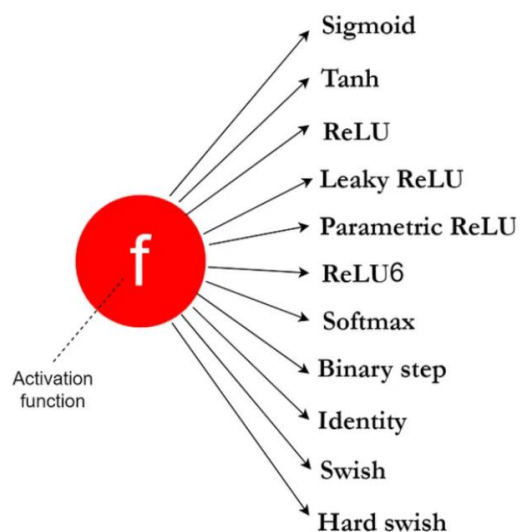
- Activation function also helps to normalize the output of **each neuron to a range between 1 and 0 or between -1 and 1.**



- In a neural network, inputs are fed into the neurons in the input layer. Each neuron has **a weight, and multiplying the input number with the weight** gives the output of the neuron, which is transferred to the next layer.
- The activation function is a mathematical “**gate**” in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold.

Neural networks use non-linear activation functions, which can help the network learn complex data, compute and learn almost any function representing a question, and provide accurate predictions.

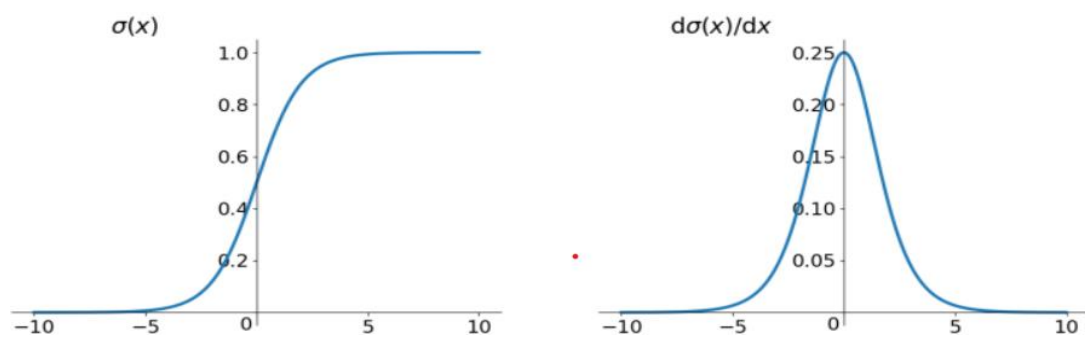
Commonly used activation functions



1. Sigmoid function

The Sigmoid function is the most frequently used activation function in the beginning of deep learning. It is a **smoothing function** that is easy to derive.

In the sigmoid function, we can see that its output is in the **open interval (0,1)**. This is very interesting. You can think of probability, but in the strict sense, don't treat it as probability. The sigmoid function was once more popular. It can be thought of as the firing rate of a neuron. In the middle where the slope is relatively large, it is the sensitive area of the neuron. On the sides where the slope is very gentle, it is the neuron's inhibitory area.



The function itself has certain defects.

- 1) When the input is slightly away from the coordinate origin, the gradient of the function becomes very small, almost zero.
- 2) In the process of neural network backpropagation, we all use the chain rule of differential to calculate the differential of each weight w . When the backpropagation passes through the sigmoid function, **the differential on this chain is very small**. Moreover, it may pass through many sigmoid functions, which will eventually cause the weight w to have little effect on the loss function, which is not conducive to the optimization of the weight. This problem is called gradient saturation or gradient dispersion.
- 3) The function output is not centered on 0, which will reduce the efficiency of weight update.
- 4) The sigmoid function performs exponential operations, which is slower for computers.

Advantages of Sigmoid Function

- Smooth gradient, preventing “jumps” in output values.
- Output values bound between 0 and 1, normalizing the output of each neuron.
- Clear predictions, i.e very close to 1 or 0.

Sigmoid has three major disadvantages

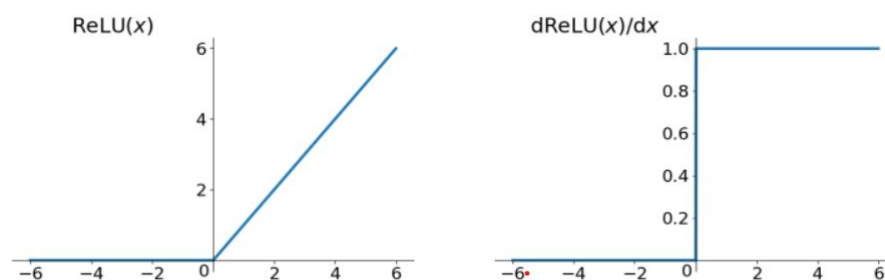
- Prone to gradient vanishing
- Function output is not zero-centered
- Power operations are relatively time consuming

2. Tanh function

- Tanh is a hyperbolic tangent function. The curves of tanh function and sigmoid function are relatively similar. Let 's compare them. First of all, when the input is large or small, the output is almost smooth and the gradient is small, which is not conducive to weight update. The difference is the output interval.
- The output interval of tanh is 1, and the whole function is 0-centric, which is better than sigmoid.

3. ReLU function

The ReLU function is actually a function that takes the maximum value. Note that this is not fully interval-derivable, but we can take sub-gradient, as shown in the figure below. Although ReLU is simple, it is an important achievement in recent years.



The ReLU (Rectified Linear Unit) function is an activation function that is currently more popular. Compared with the sigmoid function and the tanh function, it has the following advantages:

- 1) When the input is positive, there is no gradient saturation problem.
- 2) The calculation speed is much faster. **The ReLU function has only a linear relationship.** Whether it is forward or backward, it is much faster than sigmoid and tanh. (Sigmoid and tanh need to calculate the exponent, which will be slower.)

There are disadvantages:

- 1) When the input is negative, ReLU is completely inactive, which means that once a negative number is entered, ReLU will die. In this way, in the forward propagation process, it is not a problem. Some areas are sensitive and some are insensitive. But in the backpropagation process, **if you enter a negative number, the gradient will be completely zero**, which has the same problem as the sigmoid function and tanh function.
- 2) We find that the output of the ReLU function is either 0 or a positive number, which means that the ReLU function is not a 0-centric function.

4. Leaky ReLU function

- In order to solve the Dead ReLU Problem, people proposed to set the first half of ReLU 0.01x instead of 0. Another intuitive idea is a parameter-based method,

$$\text{ReLU} : f(x) = \max(\alpha, x),$$

- which alpha can be learned from back propagation. In theory, Leaky ReLU has all the advantages of ReLU, plus there will be no problems with Dead ReLU, but in actual operation, it has not been fully proved that Leaky ReLU is always better than ReLU.

5. ELU (Exponential Linear Units) function

ELU is also proposed to solve the problems of ReLU. Obviously, ELU has all the advantages of ReLU, and:

- No Dead ReLU issues
- The mean of the output is close to 0, zero-centered

6.Softmax

- Softmax is different from the normal max function: the max function only outputs the largest value, and Softmax ensures that smaller values have a smaller probability and will not be discarded directly. It is a "max" that is "soft".
- The denominator of the Softmax function combines all factors of the original output value, which means that the different probabilities obtained by the Softmax function are related to each other

7.PRelu (Parametric ReLU)

- PReLU is also an improved version of ReLU. In the negative region, PReLU has a small slope, which can also avoid the problem of ReLU death. Compared to ELU, PReLU is a linear operation in the negative region. Although the slope is small, it does not tend to 0, which is a certain advantage.
- We look at the formula of PReLU. The parameter α is generally a number between 0 and 1, and it is generally relatively small, such as a few zeros. When $\alpha = 0.01$, we call PReLU as Leaky Relu , it is regarded as a special case PReLU it.

8.Swish (A Self-Gated) Function

- Swish's design was inspired by the use of sigmoid functions for gating in LSTMs and highway networks. We use the same value for gating to simplify the gating mechanism, which is called **self-gating**.

$$y = x * \text{sigmoid}(x)$$

9.Maxout

- The Maxout activation is a generalization of the ReLU and the leaky ReLU functions. It is a learnable activation function.
- Maxout can be seen as adding a layer of activation function to the deep learning network, which contains a parameter k. Compared with ReLU, sigmoid, etc., this layer is special in that it adds k neurons and then outputs the largest activation value. Value

10.Softplus

- The softplus function is similar to the ReLU function, but it is relatively smooth. It is unilateral suppression like ReLU. It has a wide acceptance range (0, + inf).

$$f(x) = \ln(1 + \exp x)$$

Loss Functions:

A loss function is a function that compares the target and predicted output values; measures how well the neural network models the training data. When training, we aim to minimize this loss between the predicted and target outputs.

A Loss Function use some sort of loss function in the process of optimization or finding the best parameters (weights) for your data.

Type of Loss Function

estimate.

symbol	name	equation
\mathcal{L}_1	L_1 loss	$\ \mathbf{y} - \mathbf{o}\ _1$
\mathcal{L}_2	L_2 loss	$\ \mathbf{y} - \mathbf{o}\ _2^2$
$\mathcal{L}_1 \circ \sigma$	expectation loss	$\ \mathbf{y} - \sigma(\mathbf{o})\ _1$
$\mathcal{L}_2 \circ \sigma$	regularised expectation loss ¹	$\ \mathbf{y} - \sigma(\mathbf{o})\ _2^2$
$\mathcal{L}_\infty \circ \sigma$	Chebyshev loss	$\max_j \sigma(\mathbf{o})^{(j)} - \mathbf{y}^{(j)} $
hinge	hinge [13] (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)} \mathbf{o}^{(j)})$
hinge ²	squared hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)} \mathbf{o}^{(j)})^2$
hinge ³	cubed hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)} \mathbf{o}^{(j)})^3$
log	log (cross entropy) loss	$-\sum_j \mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}$
log ²	squared log loss	$-\sum_j [\mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}]^2$
tan	Tanimoto loss	$\frac{-\sum_j \sigma(\mathbf{o})^{(j)} \mathbf{y}^{(j)}}{\ \sigma(\mathbf{o})\ _2 + \ \mathbf{y}\ _2 - \sum_j \sigma(\mathbf{o})^{(j)} \mathbf{y}^{(j)}}$
D_{CS}	Cauchy-Schwarz Divergence [3]	$-\log \frac{\sum_j \sigma(\mathbf{o})^{(j)} \mathbf{y}^{(j)}}{\ \sigma(\mathbf{o})\ _2 \ \mathbf{y}\ _2}$

1. L1 and L2 loss

L_1 and L_2 are two common loss functions in machine learning which are mainly used to minimize the error

- L_1 loss function are also known as Least Absolute Deviations in short LAD.
- L_2 loss function are also known as Least square errors in short LS.

L1 Loss function

It is used to minimize the error which is the sum of all the absolute differences in between the true value and the predicted value.

L2 Loss Function

It is also used to minimize the error which is the sum of all the squared differences in between the true value and the predicted value.

2. Huber Loss

Huber Loss is often used in regression problems. Compared with L2 loss, Huber Loss is less sensitive to outliers(because if the residual is too large, it is a piecewise function, loss is a linear function of the residual).

3. Pseudo-Huber loss function

A smooth approximation of Huber loss to ensure that each order is differentiable.

4. Hinge Loss

Hinge loss is often used for binary classification problems, such as ground true: $t = 1$ or -1 , predicted value

$$y = wx + b$$

5. Cross-entropy loss

cross-entropy loss is mainly applied to binary classification problems. The predicted value is a probability value and the loss is defined according to the cross entropy. Note the value range of the above value: the predicted value of y should be a probability and the value range is $[0,1]$

Binary cross entropy

Binary cross-entropy is another special case of cross-entropy

Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value

Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.

The Binary cross-entropy using the formula

$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^N - (y_i * \log(p_i) + (1-y_i) * \log(1-p_i))$$

Here, p_i is the probability of class 1,

$(1-p_i)$ is the probability of class 0.

When the observation belongs to class 1 the first part of the formula becomes active and the second part vanishes and vice versa in the case observation's actual class are 0. This is how we calculate the Binary cross-entropy.

Binary Cross Entropy for Multi-Class classification

$$\text{logloss} = - \frac{1}{N} \sum_i^N \sum_j^M y_{ij} \log(p_{ij})$$

6.Sigmoid-Cross-entropy loss

The above cross-entropy loss requires that the predicted value is a probability. Generally, we calculate

$$\text{scores} = x * w + b.$$

Entering this value into the sigmoid function can compress the value range to (0,1).

7.Softmax cross-entropy loss

The softmax function can convert a set of fraction vectors into corresponding probability vectors. Softmax also implements a vector of 'squashes' k-dimensional real value to the [0,1] range of k-dimensional, while ensuring that the cumulative sum is 1

Optimizer

Optimizers are the extended class, which include added information to train a specific model. The optimizer class is initialized with given parameters but it is important to remember that no Tensor is needed. The optimizers are used for improving speed and performance for training a specific model.

An optimizer is an algorithm or function that adapts the neural network's attributes, like learning rate and weights.

- Hence, it assists in **improving the accuracy and reduces the total loss**.
- Optimizers help to get results faster

The change your weights or learning rates of your neural network to reduce the losses is defined by the optimizers you use. Optimization algorithms are responsible for reducing the losses and to provide the most accurate results possible

Different types of Optimizers

1. Gradient Descent
2. Stochastic Gradient Descent (SGD)
3. Mini Batch Stochastic Gradient Descent (MB-SGD)
4. SGD with momentum
5. Nesterov Accelerated Gradient (NAG)
6. Adaptive Gradient (AdaGrad)
7. AdaDelta
8. RMSprop
9. Adam

Gradient Descent:

- gradient descent is to minimize the convex function using iteration of parameter updates
- Gradient descent is an optimization algorithm used to find the values of parameters of a function that minimizes a cost function.

Stochastic Gradient Descent (SGD):

- Compared with BGD's calculation of gradients with all data at one time, SGD updates the gradient of each sample with each update.

$$x += - \text{learning_rate} * dx$$

- For large data sets, there may be similar samples, so BGD calculates the gradient. There will be redundancy, and SGD is updated only once, there is no redundancy, it is faster, and new samples can be added.

Mini Batch Stochastic Gradient Descent (MB-SGD)

- MBGD uses a small batch of samples, that is, n samples to calculate each time. In this way, it can reduce the variance when the parameters are updated, and the convergence is more stable. It can make full use of the highly optimized matrix operations in the deep learning library for more efficient gradient calculations.
- The difference from SGD is that each cycle does not act on each sample, but a batch with n samples.

Setting value of hyper-parameters: n Generally value is 50 ~ 256

SGD with momentum

- **Momentum is momentum**, which simulates the inertia of an object when it is moving, that is, the direction of the previous update is retained to a certain extent during the update, while the current update gradient is used to fine-tune the final update direction. In this way, you can increase the stability to a certain extent, so that you can learn faster, and also have the ability to get rid of local optimization.

Adaptive Gradient (AdaGrad)

- Adagrad is an algorithm for gradient-based optimization which adapts the learning rate to the parameters, using low learning rates for parameters associated with frequently occurring features, and using high learning rates for parameters associated with infrequent features.
- Adagrad proposed this problem, an algorithm that adaptively assigns different learning rates to various parameters among them. The implication is that for each parameter, as its total distance updated increases, its learning rate also slows.

AdaDelta:

- Adadelta is an extension of Adagrad and it also tries to reduce Adagrad's aggressive, monotonically reducing the learning rate.
- Adadelta we do not need to set the default learning rate as we take the ratio of the running average of the previous time steps to the current gradient.

RMSprop:

- RMSProp tries to resolve Adagrad's radically diminishing learning rates by using a moving average of the squared gradient. It utilizes the magnitude of the recent gradient descents to normalize the gradient.
- In RMSProp learning rate gets adjusted automatically and it chooses a different learning rate for each parameter.
- RMSProp divides the learning rate by the average of the exponential decay of squared gradients

Adaptive Moment Estimation (Adam)

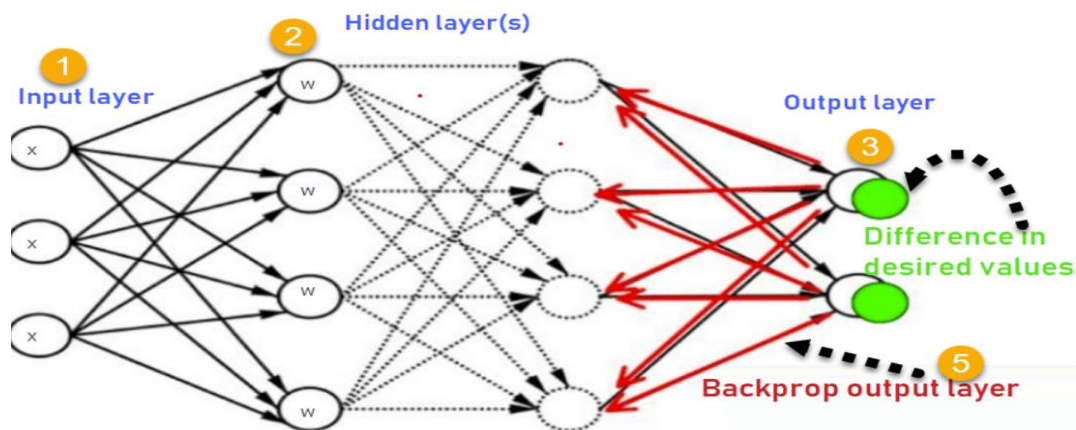
- Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop.
- Adam also keeps an exponentially decaying average of past gradients, similar to momentum.
- Adam can be viewed as a combination of Adagrad and RMSprop,(Adagrad) which works well on sparse gradients and (RMSProp) which works well in online and nonstationary settings respectively.
- Adam implements the **exponential moving average of the gradients** to scale the learning rate instead of a simple average as in Adagrad. It keeps an exponentially decaying average of past gradients.
- Adam is computationally efficient and has very less memory requirement.
- Adam optimizer is one of the most popular and famous gradient descent optimization algorithms.

Backward Propagation

Backpropagation is a process involved in training a neural network. It involves taking the error rate of a forward propagation and feeding this loss backward through the neural network layers to fine-tune the weights. Backpropagation is the essence of neural net training.

To develop a learning algorithm for multilayer feedforward neural networks, empowering the networks to be trained to capture the mapping implicitly.

The main features of Backpropagation are the **iterative, recursive and efficient method** through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained.



For back Propagation using chain rule to update the weights.

- The **chain rule** provides us a technique for finding the derivative of composite functions, with the number of functions that make up the composition determining how many differentiation steps are necessary.
- In neural networks, our main goal will be on reducing the error, to make it possible we have to update all the weights by doing backpropagation.
- We need to find a change in weights such that error should be minimum
- To do so we calculate $dE/dW1$ and $dE/dW2$.

Considering S (Summation) = $x_1W_1 + x_2W_2$

A (Activation) = sigmoid = $e^x / (1 + e^x)$

Using Chain Rule

$$\frac{dE}{dW_1} = \frac{dE}{dA} \times \frac{dA}{dS} \times \frac{dS}{dW_1}$$

$$\frac{dE}{dW_2} = \frac{dE}{dA} \times \frac{dA}{dS} \times \frac{dS}{dW_2}$$

**Forward
Propagation**



**Backward
Propagation**



Once we calculated changes in weights concerning error our next step will be on updating the weights using gradient descent procedure.

$$W_{1new} = W_{1old} - \eta \frac{dE}{dW_1}$$

$$W_{2new} = W_{2old} - \eta \frac{dE}{dW_2}$$

New weights

Forward propagation and backward propagation will be continuous for all samples until the error reaches minimum value.

Advantages of Artificial Neural Network (ANN) :

Parallel processing capability:

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

Storing data on the entire network:

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

Capability to work with incomplete knowledge:

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

Having a memory distribution:

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network :

Assurance of proper network structure:

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

Unrecognized behavior of the network:

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

Hardware dependence:

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

Difficulty of showing the issue to the network:

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

The duration of the network is unknown:

The network is reduced to a specific value of the error, and this value does not give us optimum results.

Confusion Matrix :

- The fact that it makes it easy to see whether the system is confusing two classes
- A confusion matrix is a table that is used to define **the performance of a classification algorithm**. A confusion matrix visualizes and summarizes the performance of a classification algorithm

A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes.

- The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined **if the true values for test data are known**. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix.

Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions, that are **predicted values and actual values** along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

- **True Negative:** Model has given prediction Churned, and the real or actual value was also Churned.
- **True Positive:** The model has predicted not Churned, and the actual value was also Churned.
- **False Negative:** The model has predicted Churned, but the actual value was Churned, it is also called as **Type-II error(Sensitivity)**
- **False Positive:** The model has predicted not Churned, but the actual value was Churned. It is also called a **Type-I error (specificity)**

Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix.

- **Accuracy**
- **Precision**
- **Recall**
- **F1 Score**

Accuracy:

It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Precision:

It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall:

It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall} = \frac{TP}{TP+FN}$$

F-measure:

If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$\text{F-measure} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Defining Sensitivity and Specificity

Classification models can be evaluated with the precision, recall, accuracy, and F1 metrics. We don't have to specify which group the metrics apply to because the model only has two options to choose from; either the observation belongs to the class or it does not and the model can be either correct or incorrect, hence the four portions of the confusion matrix.

Sensitivity

Sensitivity is the metric that evaluates a model's ability to predict true positives of each available category.

In addition to accuracy, sensitivity measures how many percent of the actual successes the model correctly classifies. Sensitivity uses information provided by the confusion matrix and is calculated using the formula,

Specificity

Specificity is the metric that evaluates a model's ability to predict true negatives of each available category

Specificity measures the true negative, i.e. how many percent in total the model correctly classifies the observation as a failure. Specificity also draws information provided by confusion matrix and is defined as,

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$



Data Description

Software :

Google colab:

Colab allows anybody to write and execute arbitrary python code through the browser

Which makes it a perfect tool for deep learning and data analytics enthusiasts because of computational limitations on local machines. Since a Colab notebook can be accessed remotely from any machine through a browser

Visual Studio Code

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft. Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs.

HTML and CSS

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, CSS the (visual and aural) layout, for a variety of devices.

Description of the Data

Customer Churn is one of the most important and challenging problems for businesses such as Credit Card companies, cable service providers, SASS and telecommunication companies worldwide. customer churn metrics can help businesses improve customer retention.

Because of the significant importance of customer churn within a business, stakeholders are investing more time and effort in finding out the reasoning within their organizations, how they can accurately predict the type of existing customers that can stop doing business with them and what they can [do to minimize the customer churn](#).

In our study, we will go through some Bank consumer data and see how we can leverage data insights and predictive modeling in order to improve customer retention.

The dataset for the study is called ‘**Bank Customer**’ and is downloaded from Kaggle.com (Secondary data). Originally, the data was collected from an unknown bank with customers in France, Spain and Germany, and the aim of the study was to predict customer churn.

Dataset **Link:** <https://www.kaggle.com/code/beyzanks/customer-churn-prediction-with-ann/input>

The data contains a total of 10000 observations with 14 variables, 1 dependent variable and 13 independent variables

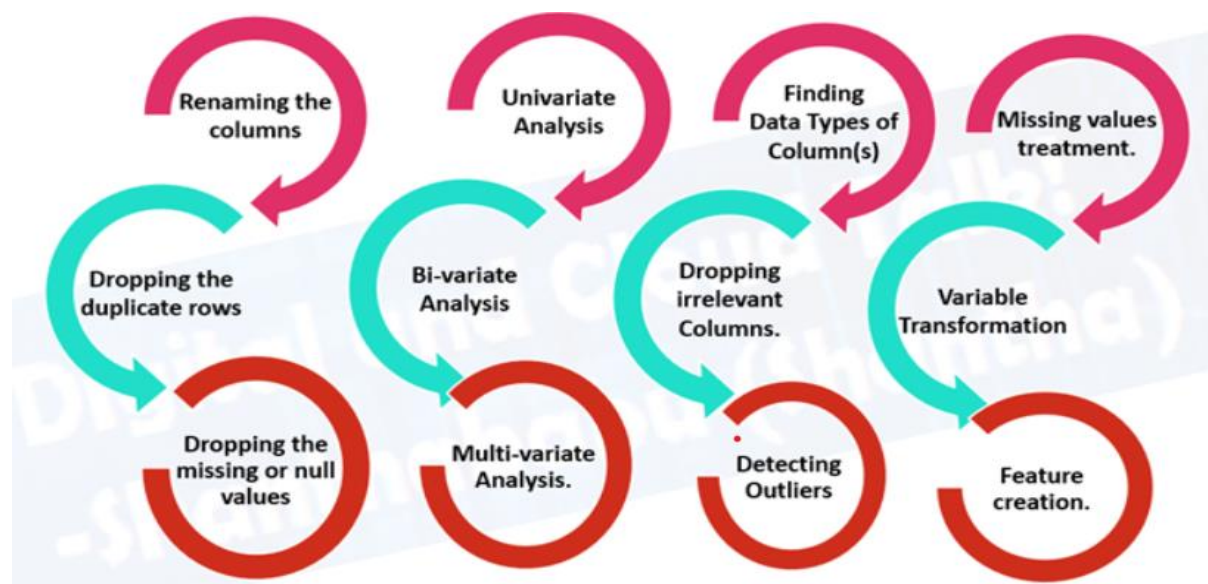
Dataset Description

Attribute	Description	Measure
Row Number	Number of customers	Nominal
Customer ID	ID of customer	Nominal
Surname	Customer name	Nominal
Credit Score	Score of credit card usage	Continuous
Geography	Location of customer	Nominal
Gender	Customer gender	Nominal
Age	Age of Customer	Continuous
Tenure	The period of having the account in months	Continuous
Balance	Customer main balance	Continuous
NumOfProducts	No of products used by customer	Discrete Variables
HasCrCard	If the customer has a credit card or not	Continuous
IsActiveMember	Customer account is active or not	Nominal
Estimated Salary	Estimated salary of the customer.	Continuous
Exited	Indicates customer leaved or not	Nominal

The dependent variable is '*Exited*'. If the dependent variable takes on a value of 1 then the customer has ended their engagement with the bank. If instead the dependent variable takes on a value of 0 the customer has not ended their engagement with the bank. In Our study, the target Variable is Binary , Our study is carry-out through Binary Classification.

Exploratory Data Analysis

Exploratory Data Analysis is a data analytics process to understand the data in depth and learn the different data characteristics, often with visual means. This allows you to get a better feel of your data and find useful patterns in it.



Below are the steps we will follow to reach our goal:

1. Importing necessary Libraries
2. Loading and Reading dataset
3. Data Understanding
4. Preprocessing
5. Visualizing dataset
6. Splitting the data into train and test
7. Building Models
8. Save the Model

Importing necessary Libraries

Pandas

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. Pandas is a library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. It has a fast and efficient DataFrame object with the default and customized indexing.

NumPy

NumPy stands for numeric python which is a package for the computation and processing of the multidimensional and single dimensional array elements. NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

Matplotlib

Human minds are more adaptive for the visual representation of data rather than textual data. It is better to represent the data through the graph where we can analyze the data more efficiently and make the specific decision according to data analysis

Seaborn

Seaborn provides many color palettes and defaults beautiful styles to make the creation of many statistical plots in Python more attractive. Seaborn library aims to make a more attractive visualization of the central part of understanding and exploring data. It is built on the core of the [matplotlib](#) library and also provides dataset-oriented APIs.

Tensorflow

TensorFlow is one of the famous deep learning framework, developed by **Google** Team. It is entirely based on Python programming language and use for numerical computation and data flow, which makes machine learning faster and easier TensorFlow can train and run the deep neural networks for image recognition, handwritten digit classification, recurrent neural network, **word embedding**, **natural language processing**, video detection, and many more.

The word TensorFlow is made by two words, i.e., Tensor and Flow

1. **Tensor** is a multidimensional array
2. **Flow** is used to define the flow of data in operation.

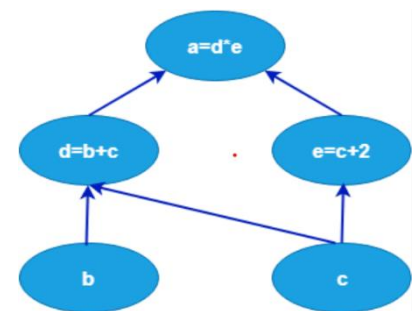
A tensor is a vector or a matrix of n-dimensional that represents all type of data. All values in a tensor hold similar data type with a known shape. The shape of the data is the dimension of the matrix or an array.

A tensor can be generated from the input data or the result of a computation. In TensorFlow, all operations are conducted inside a graph. The graph is a set of calculation that takes place successively. Each transaction is called an op node are connected.

Working of Tensorflow

TensorFlow makes use of a graph framework

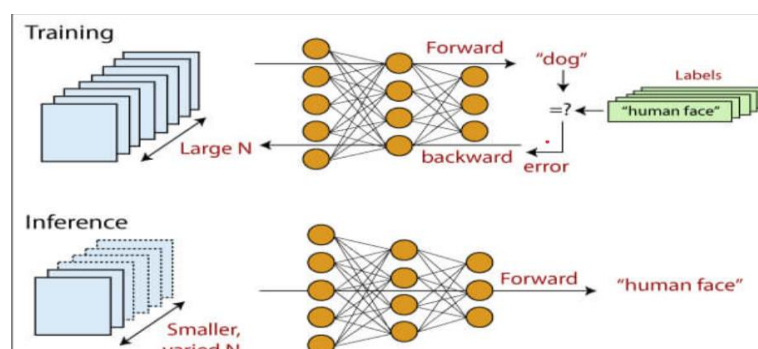
$$a = (b+c)*(c+2)$$



Features of TensorFlow

- Responsive Construct
- Flexible
- Easily Trainable
- Large Community
- Open Source
- Feature Columns

Parallel Neural Network Training



Flask

Flask is a web framework that provides libraries to build lightweight web applications in python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask is a widely used micro web framework for creating APIs in Python. It is a simple yet powerful web framework that is designed to get started quickly and easily, with the ability to scale up to complex applications.

2. Reading and Loading the data

The Customer Churn Prediction data file is first stored as CSV file format and then imported into Google Colab using pandas, its easy to read the csv file format

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

3. Basic Data Understanding:

Number of rows : 10000

Number of columns : 14

Number of missing values per
column and their percentage : 0

Total missing values and it's
percentage : 0

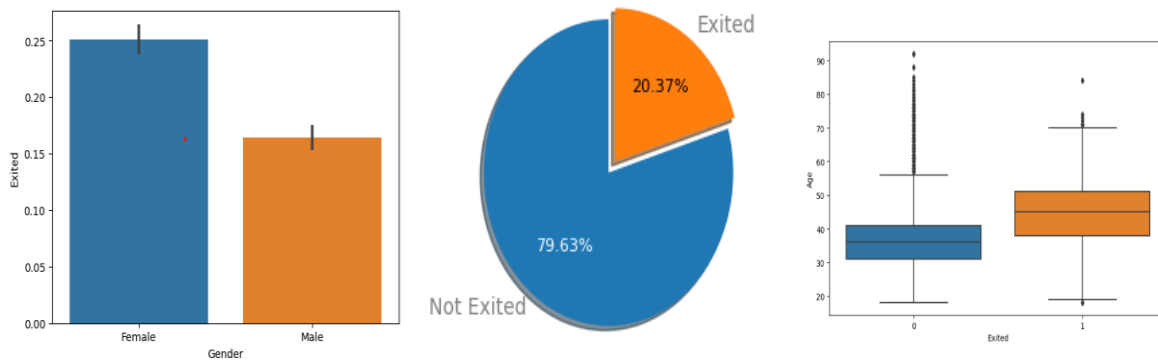
Number of categorical columns : Surname, Geography, Gender

Number of numerical columns : CreditScore, Age, Tenure, Balance,
NumOfProducts, HasCrCard, IsActiveMember,
EstimatedSalary, Exited

Number of duplicate rows : 0

"Better data beats fancier algorithms".

Visualizing dataset

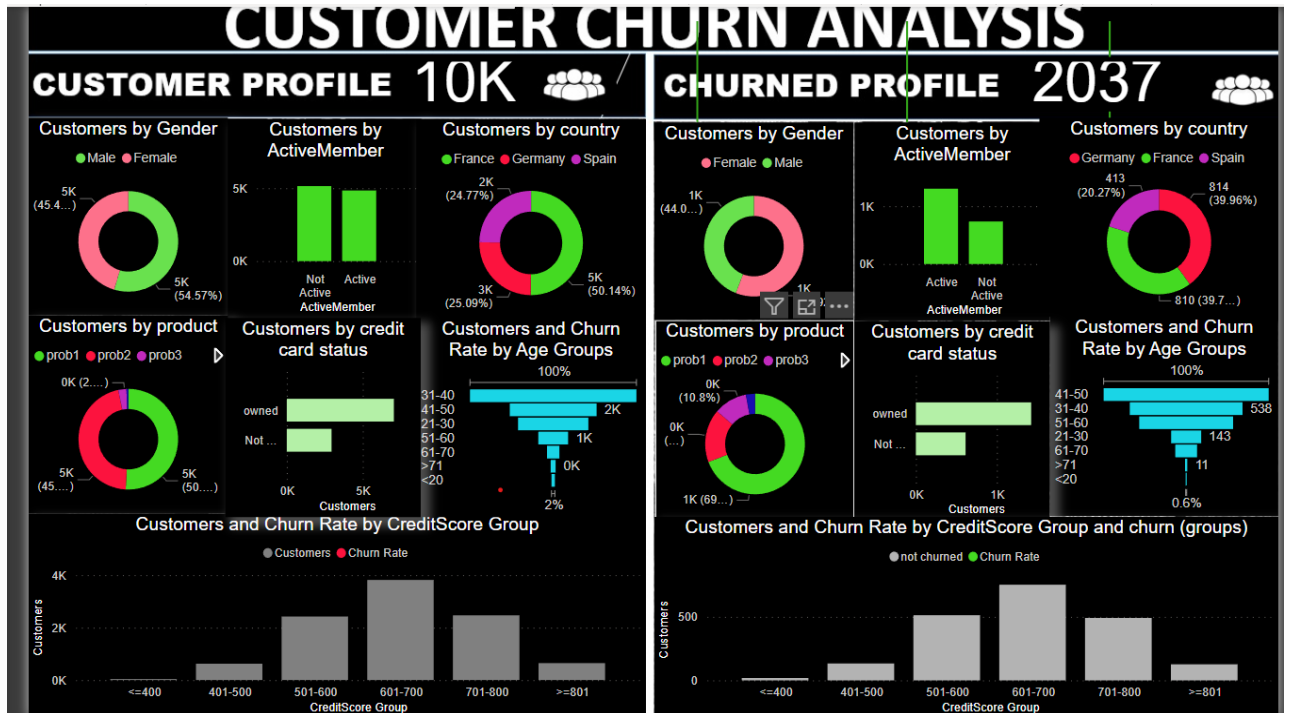


To Visualizing Our data for more Attractive and Interaction , We use the PowerBI Tool

Power BI can help connect disparate data sets, transform and clean the data into a data model and create charts or graphs to provide visuals of the data. All of this can be shared with other Power BI users within the organization.



Click **SUMMARY**, redirect to the Dashboard



The Customer Profile on left, the original data is been plotted and its easy to know the specific feature is important or not important

The Customer Churn Profile on Right, Whether the Customer are churned from a services, or we will study which age group person are churned or not. We will able to study the particular features, how the customer churned

As an Statistician able to study, how the customer are affect from various factor. Our Client is interest and solve the issue, Customer will continue the service

Click **Customer Details**, redirect to the Customers DashBoard

In this , Customers Dashboard, We select any particular **customer ID**

It Show

Personal details

Bank Detail

Exited : whether the customer is Churned or not

CUSTOMERS DASHBOARD

CUSTOMER ID

15569248

Personal Details

Surname : Milanesi1
Age : 43
Gender : Female
Geography : France

BANK DETAILS

Credit Score : 554
Balance : ₹ 0
Tenure : 10
products : prob2
Active number:Active
credit status:554

EXITED CUSTOMER

Churn : not churned

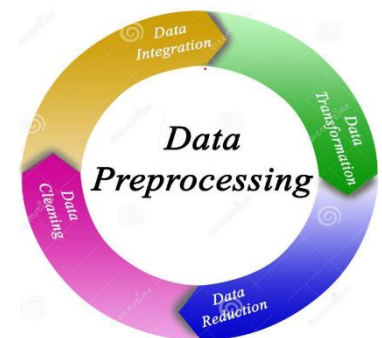
4.Data Preprocessing

Data Cleaning : Data cleaning is a crucial step in the machine learning (ML) pipeline, as it involves identifying and removing any missing, duplicate, or irrelevant data. The goal of data cleaning is to ensure that the data is accurate, consistent, and free of errors

- Missing Values

How To Handle Categorical Missing Values

- 1.Filling with Arbitrary value or Replace NaN with a Scalar Value
- 2.Filing Nan with pad/fill(Fill methods Forward) and fill/backfill(Fill methods Backward)
- 3.Drop Missing Values
- 4.statical method
 - mean
 - median
 - mode
- 5.Capturing NAN values with a new feature
- 6.End of Distribution imputation



How To Handle Categorical Missing Values

1. With Frequent Number
2. Adding a variable to capture NAN
3. Suppose if you have more frequent categories, we just replace NAN with a new category
 - **Removing duplicates:** This step involves identifying and removing any duplicate data, which can be done by using techniques such as data deduplication or data deduplication algorithms.
 - **Handling outliers:** This step involves identifying and handling any outliers in the data, which can be done by removing the outliers or transforming the data to reduce the impact of the outliers.

Outlier

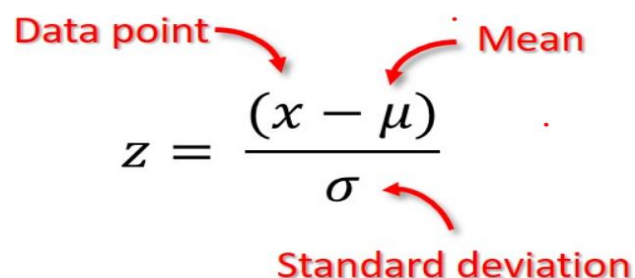
Various ways of finding the outlier.

1. Using scatter plots
2. Box plot
3. using z score
4. using the IQR Inter Quantile range

Univariate Outlier (Z-Score) :

Z score is an important concept in statistics. **Z score is also called standard score.** This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean.

If the z score of a data point is more than 3, it indicates that the data point is quite different from the other data points. Such a data point can be an outlier.



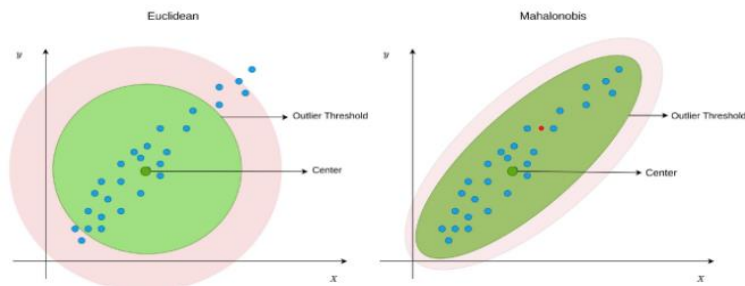
The diagram shows the Z-score formula with red arrows pointing to its components:
$$Z = \frac{(x - \mu)}{\sigma}$$
 Data point points to x , **Mean** points to μ , and **Standard deviation** points to σ .

Multivariate Outlier (Mahalanobis distance) :

The Mahalanobis distance has a major application for the detection of outliers

Mahalanobis distance measures the number of standard deviations that an observation is from the mean of a distribution.

For a multivariate data, it uses a covariance matrix of variables to find the distance between data points and the center



Which is the distance of a data point from the calculated centroid of the other cases where the centroid is calculated as the intersection of the mean of the variables being assessed.

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

where:

- \vec{x} is the **observation** to find its distance;
- $\vec{\mu}$ is the **mean** of the observations;
- S is the [Covariance Matrix](#).

Feature Engineering

Feature Engineering is the process of selecting, Creating and transforming features in a dataset to improve the performance of machine learning models

Feature Engineering is Important

Better model accuracy: Feature engineering can help to identify the most important features and remove noise, leading to a more accurate and reliable model.

Improved interpretability: Feature engineering can help to create features that are more easily interpretable, making it easier to understand how the model is making predictions.

Reduced overfitting: Feature engineering can help to reduce overfitting by creating features that are more generalizable and less dependent on specific instances in the training data

Increased model robustness: Feature engineering can help to make the model more robust to changes in the data or new data inputs, improving its generalization capabilities

- First, In our study selecting the require feature
- In Our data categorical features is covert into a Numerical features using
One-Hot Encoding
Label Encoding
get_dummies

Get_Dummies

The get_dummies function is used to convert categorical variables into dummy or indicator variables. A dummy or indicator variable can have a **value of 0 or 1**.

```
pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None)
```

Correlation analysis

Correlation analysis is a statistical method used to measure the strength of the linear relationship between two variables and compute their association. Correlation analysis calculates the level of change in one variable due to the change in the other.

correlation analysis to analyze

- to identify relationships,
- patterns,
- significant connections,
- trends between two variables or datasets

Value of r	Correlation
• 1.00	Perfect or ideal
• 0.90 -- 0.99	Excellent
• 0.80 -- 0.89	Very high correlation
• 0.60 -- 0.79	High correlation
• 0.40 -- 0.59	Medium correlation
• 0.20 -- 0.39	Low correlation
• 0.00 -- 0.19	Negligible correlation
• 0.00	No correlation

In terms of the strength of the relationship, the correlation coefficient's value varies between +1 and -1. A value of ± 1 indicates a perfect degree of association between the two variables.

Heat Map

A HeatMap is a graphical representation of data that uses a system of color-coding to represent different values

Heatmaps are used to show relationships between two variables, one plotted on each axis. By observing how cell colors change across each axis, you can observe if there are any patterns in value for one or both variables.

The major benefit of heatmap visualization is that it enables data to be presented visually which allows us to easily consume information and make more sense of it.

A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.

PairPlot

A Pairplot plot a pairwise relationships in a dataset. The pairplot function creates a grid of Axes such that each variable in data will be shared in the y-axis across a single row and in the x-axis across a single column.

It provides a high-level interface for drawing attractive and informative statistical graphics

The Pairplot function allows the users to create an axis grid via which each numerical variable stored in data is shared across the X- and Y-axis in the structure of columns and rows. We can create the [Scatter plots in order to display the pairwise relationships](#) in addition to the distribution plot displaying the data distribution in the column diagonally.

Multicollinearity

Multicollinearity is a statistical concept where several independent variables in a model are correlated. Multicollinearity refers to a situation in which [more than two explanatory variables](#) in a Multiple Regression Model are highly linearly related. There is perfect multicollinearity if, for example as in the equation above, the correlation between two independent variables equals 1 or -1 .

A simple method to detect multicollinearity in a model is by using something called the [variance inflation factor or the VIF](#) for each predicting variable.

VIF determines the strength of the correlation between the independent variables. It is predicted by taking a variable and regressing it against every other variable.

Consider a VIF (Variance Inflation Factor) > 10 as an indicator of multicollinearity, but some choose a more conservative threshold of 5

VIF-values are below 10, indicating no issues of multicollinearity, except for the interaction term. Regarding the interaction term it has a higher VIF-value, hence indicating multicollinearity. However, this can be ignored, firstly because the estimated model includes the factors of the interaction term and will therefore correlate with these. Secondly, since multicollinearity increases the variance which affects the sign and value for the estimated coefficient. The implications of multicollinearity are thus not as large for prediction modelling as for inferential purposes.

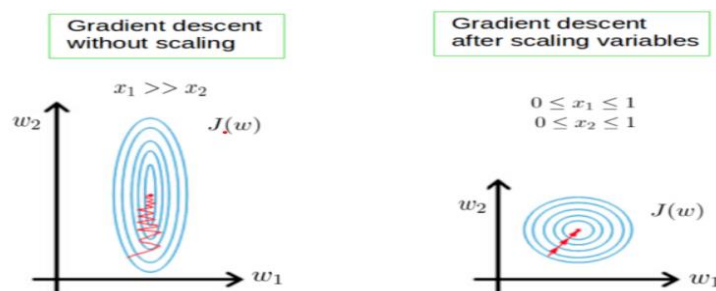
Feature Scaling

Feature scaling is nothing other than transforming the numerical features into a small range of values. Its name implies that only features are scaled. Labels or output data don't need to be scaled. Feature Preprocessing is one of the most crucial steps in building a Machine learning model. The values in each of the features plays a significant role while building ML model. For categorical data, we can encode them in different ways.

Handling continuous variables is very important as affects model performance and model building. For an example, features (columns) like income which can range from 20,000 to 100,000, and even more; while an age column which can range from 0 to 100(at the most). Thus, Income is about 1,000 times larger than age.

But how can we be sure that the model treats both these variables equally? When we feed these features to the model as it is, **there is every chance that the income will influence the result more due to its larger value.** But this doesn't necessarily mean it is more important as a predictor. So, to give importance to both Age, and Income, we need feature scaling.

If we don't do feature scaling then the machine learning model gives higher weightage to higher values and lower weightage to lower values. Also, takes a lot of time for training the machine learning model.



Different Scaling techniques

1. MinMax Scaler
2. Standard Scaler
3. Robust Scaler
4. MaxAbsScaler
5. Power Transformer Scaler
6. Box-Cox transform

7. The Yeo-Johnson transform
8. Unit Vector Scaler/Normalizer
9. Custom Transformer

Standardization

In standardization, the numerical features are rescaled to have the 0 mean (μ) and unity standard deviation ($\sigma = 1$).

Here is the formula of standardization

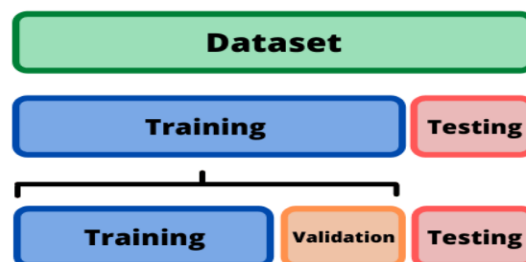
$$X_{std} = \frac{X - \mu}{\sigma}$$

The standardization scaling technique is suitable for data that has a normal or gaussian distribution. Some machine learning models such as support vector machines (with Radial Basis Function (RBF) kernel) and linear models (linear and logistic regression) expect the input data to have a normal distribution.

1. Normalization is scaling the data to be between 0 and 1. It is preferred when the data doesn't have a normal distribution.
2. Standardization is scaling the data to have 0 mean and unit standard deviation. It is preferred when the data has a normal or gaussian distribution.

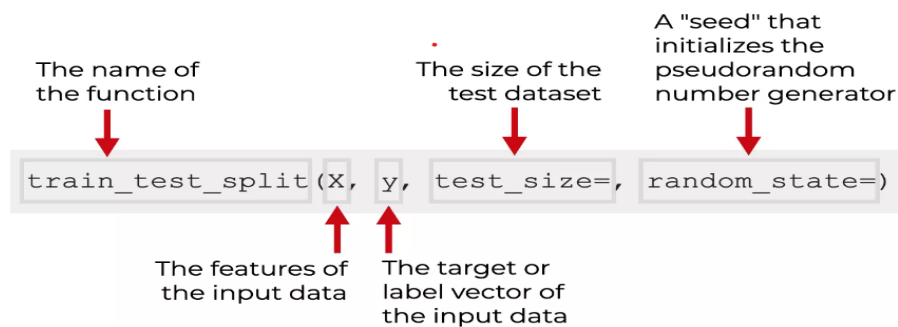
Train and Test

A train test split is when you split your data into a training set and a testing set. The training set is used for training the model, and the testing set is used to test your model. This allows you to train your models on the training set, and then test their accuracy on the unseen testing set.



Train_Test_Split

The **Sklearn** `train_test_split` function helps us create our training data and test data. This is because typically, the training data and test data come from the same original dataset. To get the data to build a model, we start with a single dataset, and then we split it into two datasets: train and test



TEST_SIZE

The `test_size` parameter enables you to specify the **size of the output test set**. If the argument is an integer, the size of the test set will be equal to that number. If the argument is a float, it must be between 0 and 1, and the number will represent the proportion of observations that will be in the test set.

RANDOM_STATE

The `random_state` parameter controls how the pseudo-random number generator randomly selects observations to go into the training set or test set.

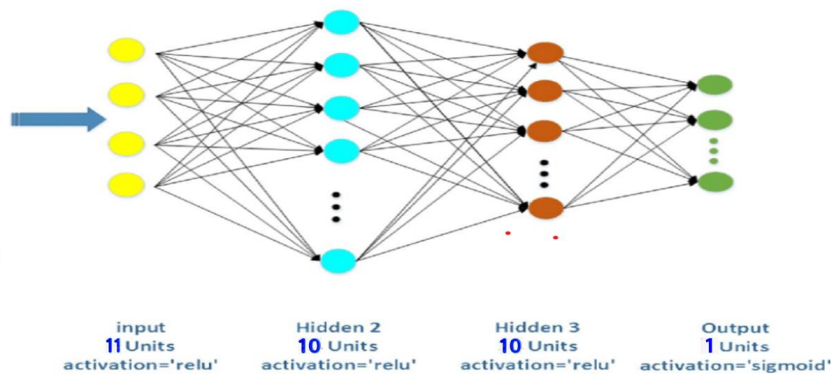
The `train_test_split` will shuffle the data in the same order prior to the split, every time you use the function with that same integer.

Effectively, if you provide an integer to this parameter, it will make your code exactly reproducible across every call of the function.

Model : Artificial Neural Network (ANN)

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes

```
1 classifier=Sequential()  
2 classifier.add(Dense(10,input_shape=(11,),activation='relu'))  
3 classifier.add(Dense(units=10,activation='relu'))  
4 classifier.add(Dense(units=10,activation='relu'))  
5 classifier.add(Dense(1,activation='sigmoid'))  
6 classifier.summary()
```



Sequential

The sequential model allows us to specify a neural network, precisely, sequential: from input to output, passing through a series of neural layers, one after the other. The ANN also has layers in sequential order and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.

Types of sequence models

Sequence-to-one: In sequence-to-one sequence model, the input data is sequence and output data is non sequence.

Sequence-to-sequence: In sequence-to-sequence sequence model, the input data is sequence and output data is sequence.

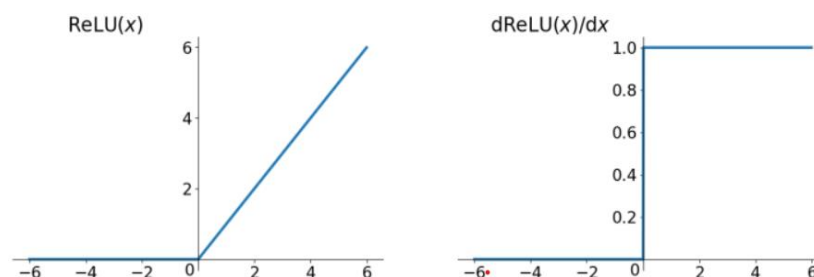
Dense

A dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. A Dense layer feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer

Activation Function

ReLU function

The ReLU function is actually [a function that takes the maximum value](#). Note that this is not fully interval-derivable, but we can take sub-gradient, as shown in the figure below. Although ReLU is simple, it is an important achievement in recent years.



The ReLU (Rectified Linear Unit) function is an activation function that is currently more popular. Compared with the sigmoid function and the tanh function, it has the following advantages:

- 1) When the input is positive, there is no gradient saturation problem.
- 2) The calculation speed is much faster. The ReLU function has only a linear relationship. Whether it is forward or backward, it is much faster than sigmoid and tanh. (Sigmoid and tanh need to calculate the exponent, which will be slower.)

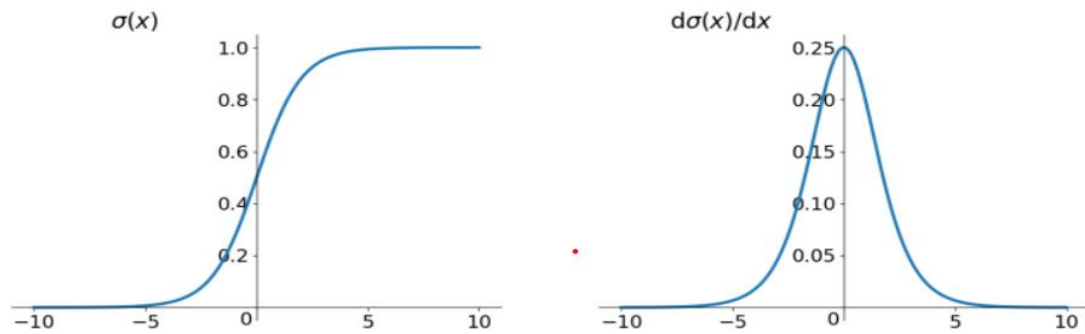
There are disadvantages:

- 1) When the input is negative, ReLU is completely inactive, which means that once a negative number is entered, ReLU will die. In this way, in the forward propagation process, it is not a problem. Some areas are sensitive and some are insensitive. But in the backpropagation process, if you enter a negative number, the gradient will be completely zero, which has the same problem as the sigmoid function and tanh function.
- 2) We find that the output of the ReLU function is either 0 or a positive number, which means that the ReLU function is not a 0-centric function.

Sigmoid function

The Sigmoid function is the most frequently used activation function in the beginning of deep learning. It is a smoothing function that is easy to derive.

In the sigmoid function, we can see that its output is in the open interval (0,1). This is very interesting. You can think of probability, but in the strict sense, don't treat it as probability. The sigmoid function was once more popular. It can be thought of as the firing rate of a neuron. In the middle where the slope is relatively large, it is the sensitive area of the neuron. On the sides where the slope is very gentle, it is the neuron's inhibitory area.



The function itself has certain defects.

- 1) When the input is slightly away from the coordinate origin, the gradient of the function becomes very small, almost zero.
- 2) In the process of neural network backpropagation, we all use the chain rule of differential to calculate the differential of each weight w . When the backpropagation passes through the sigmoid function, the differential on this chain is very small. Moreover, it may pass through many sigmoid functions, which will eventually cause the weight w to have little effect on the loss function, which is not conducive to the optimization of the weight. This problem is called gradient saturation or gradient dispersion.
- 3) The function output is not centered on 0, which will reduce the efficiency of weight update.
- 4) The sigmoid function performs exponential operations, which is slower for computers.

Advantages of Sigmoid Function :

- Smooth gradient, preventing “jumps” in output values.
- Output values bound between 0 and 1, normalizing the output of each neuron.
- Clear predictions, i.e very close to 1 or 0.

Sigmoid has three major disadvantages:

- Prone to gradient vanishing
- Function output is not zero-centered

- Power operations are relatively time consuming

Classifier Summary

It contain all the parameter and we initialize the Number of Nodes,Hidden Layers, Output and it give summary of the How may Layer are present in the model

```
classifier.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 10)	120
dense_1 (Dense)	(None, 10)	110
dense_2 (Dense)	(None, 10)	110
dense_3 (Dense)	(None, 1)	11

```

-----
Total params: 351
Trainable params: 351
Non-trainable params: 0
-----

```

Optimizers

Adaptive Moment Estimation (Adam)

- Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelata and RMSprop.
- Adam also keeps an exponentially decaying average of past gradients, similar to momentum.
- Adam can be viewed as a combination of Adagrad and RMSprop,(Adagrad) which works well on sparse gradients and (RMSProp) which works well in online and nonstationary settings repectively.
- Adam implements the **exponential moving average of the gradients** to scale the learning rate instead of a simple average as in Adagrad. It keeps an exponentially decaying average of past gradients.
- Adam is computationally efficient and has very less memory requirement.
- Adam optimizer is one of the most popular and famous gradient descent optimization algorithms

Learning Rate

The learning rate is the most important neural network hyperparameter. It can decide many things when training the network. In most optimizers in Keras, the default learning rate value is 0.001. It is the recommended value for getting started with training.

The learning rate. **Defaults to 0.001**. beta_1: A float value or a constant float tensor, or a callable that takes no arguments and returns the actual value to use.

```
1 opt=tensorflow.keras.optimizers.Adam(learning_rate=0.01)
2 classifier.compile(optimizer=opt,loss='binary_crossentropy',metrics=['accuracy'])
3 import tensorflow as tf
4 early_stopping=tf.keras.callbacks.EarlyStopping(
5     monitor="val_loss",
6     patience=5
7 )
8 )
9 model_history=classifier.fit(X_train,y_train,validation_data=0.20,batch_size=10,epochs=100,callbacks=early_stopping)
```

Loss Function

Binary cross entropy

Binary cross-entropy is another special case of cross-entropy

Binary cross entropy **compares each of the predicted probabilities to actual class output which can be either 0 or 1**. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value

Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.

The Binary cross-entropy using the formula

$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^N - (y_i * \log(p_i) + (1-y_i) * \log(1-p_i))$$

Here, p_i is the probability of class 1,

(1-pi) is the probability of class 0.

When the observation belongs to class 1 the first part of the formula becomes active and the second part vanishes and vice versa in the case observation's actual class are 0. This is how we calculate the Binary cross-entropy.

Binary Cross Entropy for Multi-Class classification

$$\text{logloss} = - \frac{1}{N} \sum_i^N \sum_j^M y_{ij} \log(p_{ij})$$

Cross Validation

Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set. The three steps involved in cross-validation are as follows :

1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.





DATA ANALYSIS AND INTERPRETATION

1. Basic data Understanding

number of rows : 10000

number of columns : 14

Number of missing values per column: 0

Total missing values: 0

Number of duplicate rows : 0

Number of categorical columns : Surname, Geography, Gender

Number of numerical columns : CreditScore, Age, Tenure, Balance,

NumOfProducts, HasCrCard, IsActiveMember,

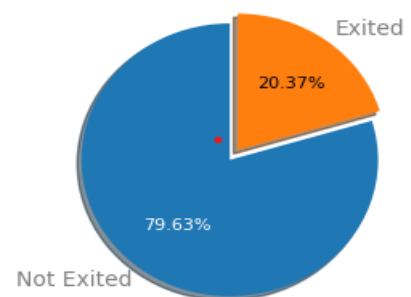
EstimatedSalary, Exited

2. visualizing the Data

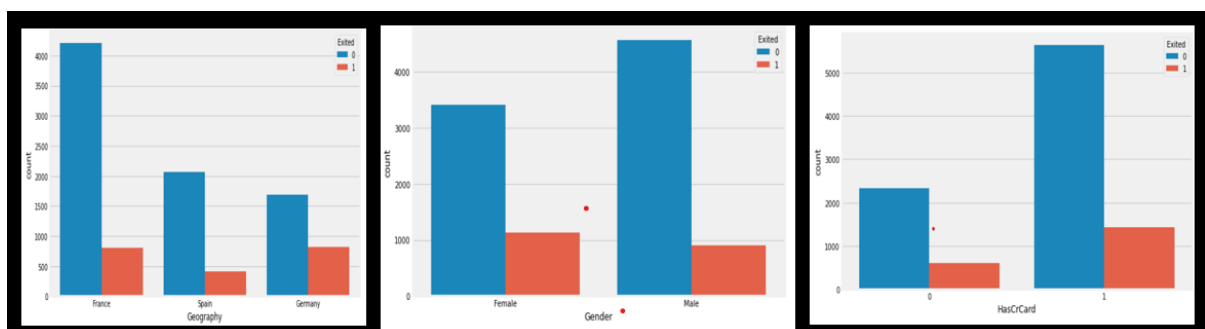
There is 20%(2037) of customer are churned from the Bank.

Remaining 80% of customer are continue the service,

For Geography features, there only 3 Unique Value ['France', 'Spain', 'Germany'],



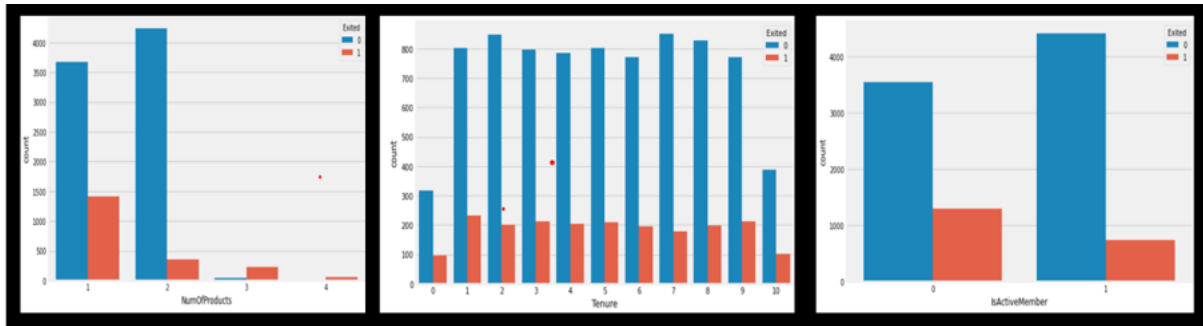
Using Barplot, we will study France and Germany are churned more than Spain .



In Gender, there are two unique value ['Male' , 'Female'] ,

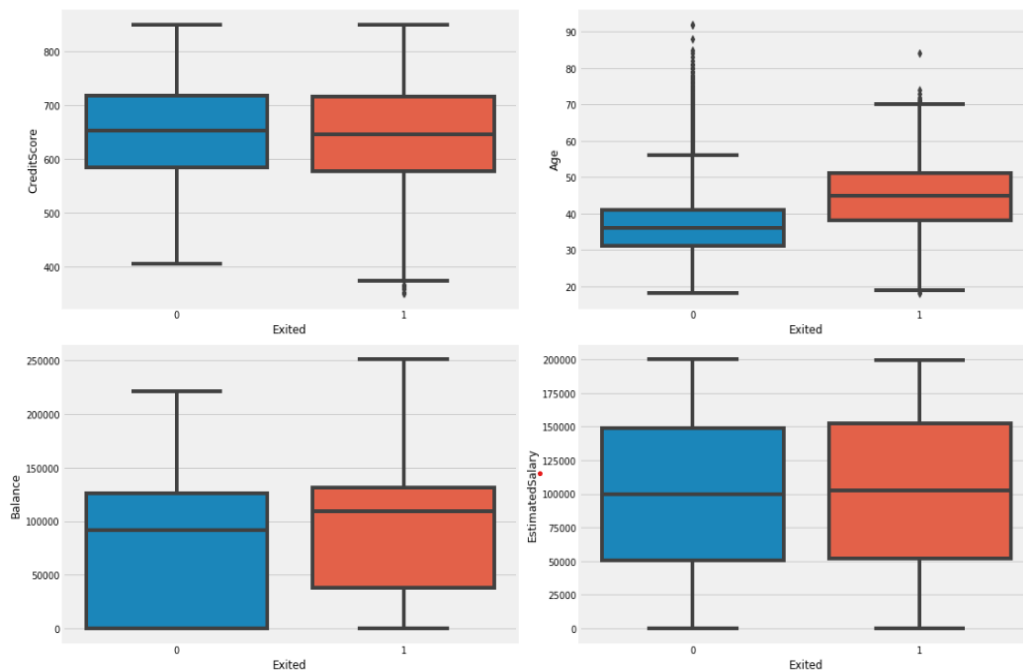
Female are little more churned than male

In Has Credit card features, 1 representing the credit card holder are Churned more than others



Observation :

- The Client more Concentrate to the Geography, as France and Germany Country are Churned more, The Client can give new Offer, Or some discounts to the Customer.
- As compatibility Female are little more Churned than Males
- If, Credit card Holder are Churned, the Service is not Good, So the Client Should Change the Service, or give Special Discount from a Credit Card Holder.
- If Number of Product is 1 , the Customer are not ready to buy the products, If the Customer are brought the 1-Product those Customers are Churned.
- Tenure looking Normal, and if non-Active Customer are Churned



Observation :

- There is no significant difference in credit score, Estimated Salary, bank balance distribution between customers which are churned or not.
- The older customers are churning more than younger ones
- Interestingly, majority of customers that churned are those with credit cards but this can be a coincidence as majority of customers have credit cards.
- the inactive members have a greater churn and the overall proportion of inactive members is also very high.

VIF(Variance inflation factor)

	VIF Factor	features
0	21.4	CreditScore
1	14.0	Age
2	3.9	Tenure
3	3.2	Balance
4	7.8	NumOfProducts
5	3.3	HasCrCard
6	2.1	IsActiveMember
7	3.9	EstimatedSalary
8	1.5	Exited
9	1.8	Germany
10	1.5	Spain
11	2.2	Male

	VIF Factor	features
0	10.6	Age
1	3.7	Tenure
2	3.0	Balance
3	6.5	NumOfProducts
4	3.2	HasCrCard
5	2.1	IsActiveMember
6	3.7	EstimatedSalary
7	1.5	Exited
8	1.8	Germany
9	1.5	Spain
10	2.2	Male

Using the VIF to detect the Multicollinearity of independent variable and we fix the cutoff value as 10, where 10 is less than any feature, we simply remove that particular feature in the data

Here $10 < 21.4$, we simply remove the **Credit Score** Feature in the data and recheck the VIF, if the feature are Multicollinearity

Outlier Detection:

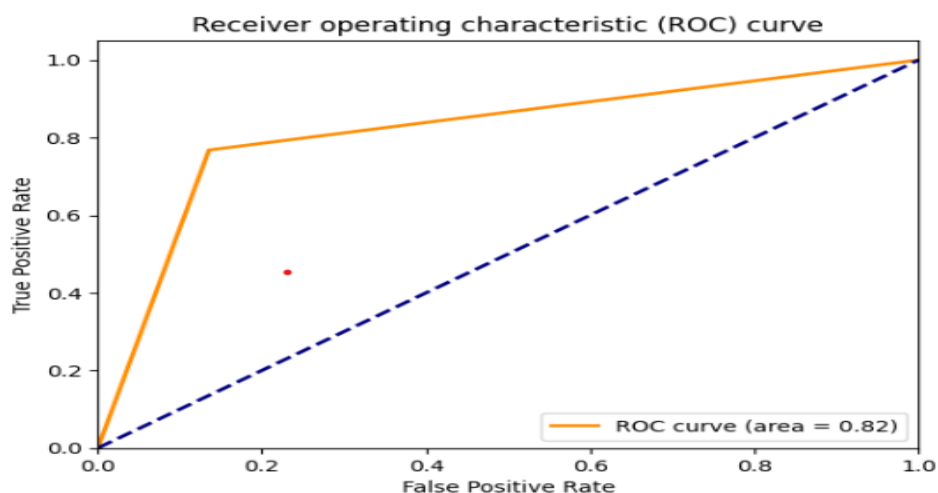
1. Univariate Outlier Detection :

For Univariate Outlier Detection , In our Study we use **Z SCORE** for detecting the outlier

```
1 Total number of outliers Age are [82, 88, 85, 84, 84, 81, 92, 92, 81, 81, 81, 83]
2
3 data = print(np.where(X['Age']>=80))
4 data = X.drop(X.index[[ 310, 766, 2458, 3033, 3387, 3531, 3994, 4931, 6443, 6759, 7526,
5 | 7956, 9080, 9309, 9490]])
6 data.shape
7 (9985, 15)
```

In our Dataset, Age column having the outlier, So we remove the Indexes more than age is 80 and we lost only the 15 records . The final Data contain 9985 * 15 are used to predict the Model

```
1 [[1570 246]
2  [ 42 139]]
3 Accuracy : 0.85578367551327
4 Sensitivity : 0.8645374449339207
5 Specificity : 0.7679558011049724
6 precision recall f1-score support
7
8 False 0.97 0.86 0.92 1816
9 True 0.36 0.77 0.49 181
10
11 accuracy 0.86 1997
12 macro avg 0.67 0.82 0.70 1997
13 weighted avg 0.92 0.86 0.88 1997
14
15 0.85578367551327
```



2. Multivariate Outlier Detection

For Multivariate Outlier Detection , In our Study we use **Mahalanobis Distance** for detecting the outlier

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_Germany	Geography_Spain	Gender_Male	mahalanobis	p
0	619	42	2	0.00	1	1	1	101348.88	0	0	0	7.750429	0.735470
1	608	41	1	83807.86	1	0	1	112542.58	0	1	0	10.405960	0.494282
2	502	42	8	159660.80	3	1	0	113931.57	0	0	0	20.987203	0.033504
3	699	39	1	0.00	2	0	0	93826.63	0	0	0	9.286469	0.595468
4	850	43	2	125510.82	1	1	1	79084.10	0	1	0	12.764869	0.308971

```
newdf = final_dataset[(final_dataset['p'] >= 0.001) ]
newdf.shape
```

(9968, 13)

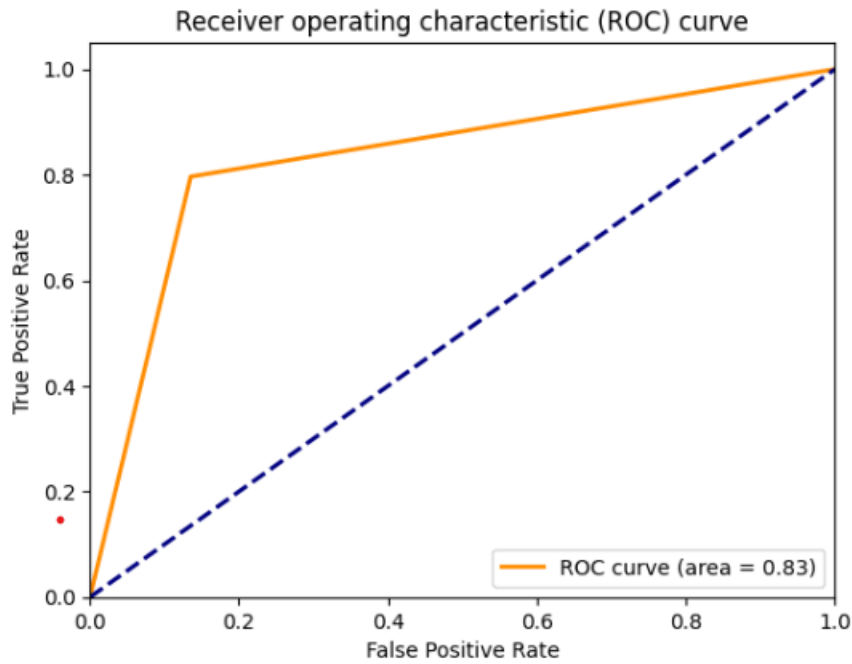
For Mahalanobis Distance we calculate the Mahalanobis Value and chi Square we calculate the P_Value,

The p-value that is **less than .001** is considered to be an outlier. We can see that the second observation is an outlier in the dataset because it has a p-value less than .001.

We get Outlier has 32 * 13

Here, The final Data contain 9968 * 13 are used to predict the Model

```
1  [[1529  46]
2  [ 239 180]]
3  Accuracy : 0.8570712136409228
4  Sensitivity : 0.9707936507936508
5  Specificity : 0.4295942720763723
6  classification report
7  _____
8  |               precision    recall  f1-score   support
9  |
10 |  False           0.97       0.86       0.91       1768
11 |  True            0.43       0.80       0.56        226
12 | _____
13 | accuracy                0.86       1994
14 | macro avg              0.70       0.83       0.74       1994
15 | weighted avg           0.91       0.86       0.87       1994
16 |
17 | 0.8570712136409228
```



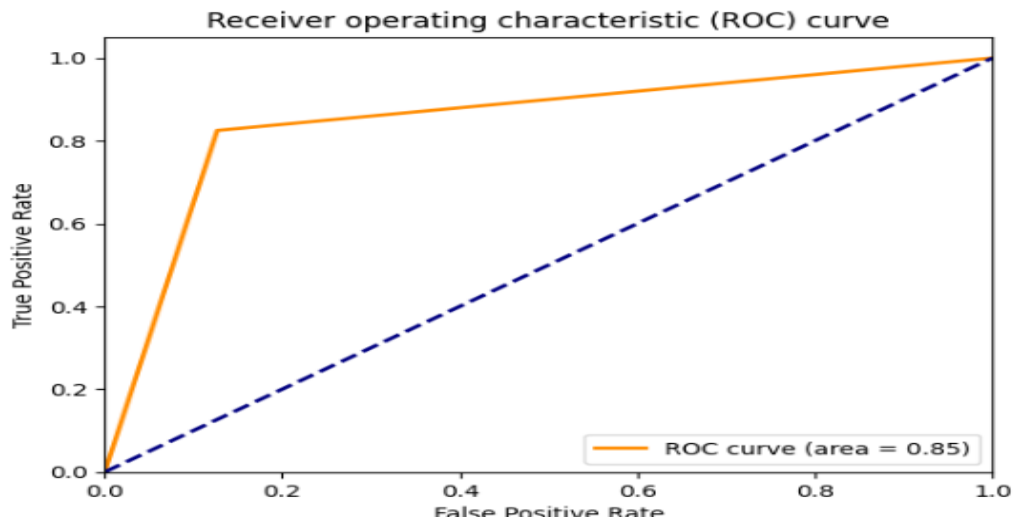
3. Without Using outlier and feature selection

Here, We done Simple ANN Model With out using any feature selection involves, In this study we use the Activation function as Relu and optimizer as Adam, Loss function is Brinary Crossentropy.

```

1  [[1557  38]
2  [ 226 179]]
3  Accuracy : 0.868
4  Sensitivity : 0.9761755485893417
5  Specificity : 0.4419753086419753
6  classification report
7  precision    recall  f1-score   support
8
9      False      0.98      0.87      0.92      1783
10     True       0.44      0.82      0.58       217
11
12    accuracy          0.87      2000
13    macro avg       0.71      0.85      0.75      2000
14    weighted avg    0.92      0.87      0.88      2000
15
16  0.868

```



4. Hyper Parameter Tunning

To Improve the Accuracy, sensitivity and Specificity , Play with different parameter, Change the Learning rate, Optimizer, Loss Function , Activation function , to get the best Accuracy .

```

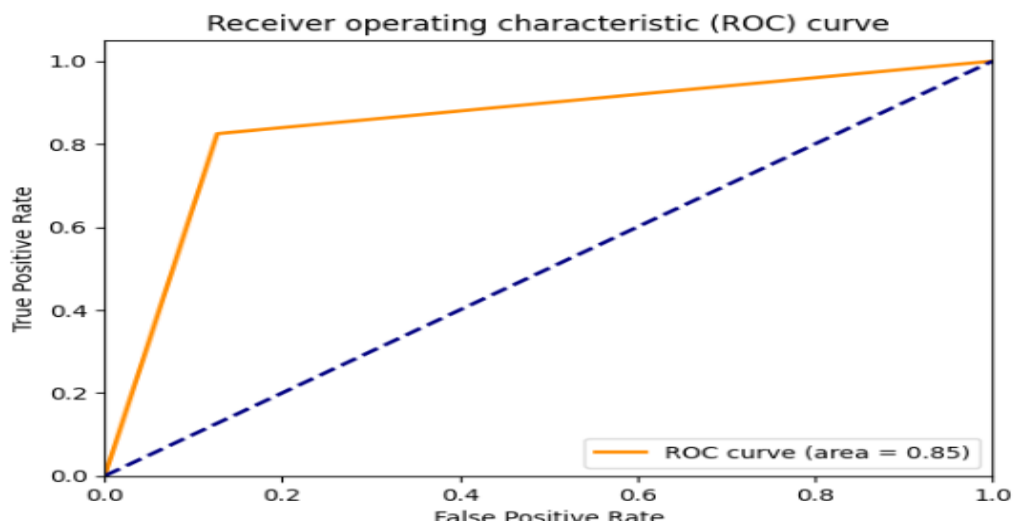
1  def build_model(hp):
2      model = keras.Sequential()
3      for i in range(hp.Int('num_layers', 2, 20)):
4          model.add(layers.Dense(units=hp.Int('units_' + str(i),
5          min_value=32,
6          max_value=512,
7          step=32),
8          activation='relu'))
9          model.add(layers.Dense(1, activation='sigmoid'))
10     model.compile(
11         optimizer=keras.optimizers.Adam(
12             hp.Choice('learning_rate', [1e-2, 1e-3, 1e-4])),
13         loss='binary_crossentropy',
14         metrics=['accuracy'])
15     return model
16
17     tuner = RandomSearch(
18         build_model,
19         objective='accuracy',
20         max_trials=5,
21         executions_per_trial=3,
22         directory='project',
23         project_name='Churn Prediction')
24     tuner.search_space_summary()

```

```

1  [[1529  66]
2  [ 201 204]]
3  Accuracy : 0.8665
4  Sensitivity : 0.9586206896551724
5  Specificity : 0.5037037037037037
6  classification report
7  _____
8  precision    recall  f1-score   support
9
10 False       0.96     0.88     0.92     1730
11 True        0.50     0.76     0.60      270
12
13 accuracy    0.87     2000
14 macro avg   0.73     0.82     0.76     2000
15 weighted avg 0.90     0.87     0.88     2000
16
17 0.8665

```



Observation

1. In our study, Using **Univariate Outlier Detection (Z-Score)** gives the better Accuracy, Specificity, Sensitivity
2. Depend on this Accuracy, We use this Model to deploy and create the User Interactive Web Application.

ACCURACY	85 %
SPECIFICITY	86%
SENSITIVITY	76%

WEB APPLICATION

This **User Interface** may help Bank Manager, how the customers are churned and what factor can easily Identified and develop the marketing actions to retain their valuable clients. Thus, due to existing customers are retained, it will provide banks with increased profits and revenues.

The screenshot shows a web browser window with the title "Customer Churn Prediction". The URL is "127.0.0.1:5000/predict". The browser's address bar shows several tabs, including "avnyaday/machine_learning_pro...", "Customer Churn Prediction", and others. The web application has a blue header with the word "BANK" on the left and navigation links "Home", "Not Banking", "Predictions", and "Contact" on the right. The main content area is a white form titled "Enter the Information Below and find if a Customer will Churn out or not". The form contains several input fields and dropdown menus, each with a red asterisk indicating a required field. The inputs are: "Credit Score" (670), "Age" (55), "Tenure" (3), "Enter the Account Balance" (50000), "Number of Products" (1), "Do the Customer have Credit Card? (1=Yes, 0=No)" (0), "Is the Customer Active Member (1=Yes, 0=No)" (0), "Enter the Estimated Salary" (55548), "Enter The Location" (Germany), and "Gender of Customer" (Male). Below the form is a blue button that says "Predict whether the Customer will leave the bank or not?". At the bottom of the form, a blue banner displays the prediction: "The Customer will leave the bank". The footer of the application says "Developed by SADISH P". The browser's taskbar at the bottom shows the Windows logo, search icon, and various application icons. The system tray on the right shows the date and time as "12-04-2023 05:04".

avnyaday/machine_learning_pro... x Customer Churn Prediction x +

127.0.0.1:5000/predict

GitHub - sanneabhi... GitHub - avnyaday/... Full Stack Data Scie... sadishpitchandi harshsanghi41 (Ha... DeepakRautella/ins... class task_MongoD... Modified Version of... keras_tuner_sample... 001_FSDS_Nov_202...

BANK Home Not Banking Predictions Contact

Enter the Information Below and find if a Customer will Churn out or not

Credit Score* 670

Age* 55

Tenure* 3

Enter the Account Balance:* 50000

Number of Products* 1

Do the Customer have Credit Card? (1=Yes, 0=No)* 0

Is the Customer Active Member (1=Yes, 0=No)* 0

Enter the Estimated Salary:* 55548

Enter The Location:* Germany

Gender of Customer:* Male

Predict whether the Customer will leave the bank or not?

The Customer will leave the bank

Developed by SADISH P

25°C Partly cloudy

ENG IN 12-04-2023 05:04

Customer will not Leave the Bank for these Specific Value

Enter the Information Below and find if a Customer will Churn out or not

Credit Score* <input type="text" value="780"/>	Is the Customer Active Member (1=Yes, 0=No)* <input type="text" value="1"/>
Age* <input type="text" value="70"/>	Enter the Estimated Salary: * <input type="text" value="55548"/>
Tenure* <input type="text" value="12"/>	Enter The Location: * <input type="text" value="Germany"/>
Enter the Account Balance: * <input type="text" value="20000"/>	Gender of Customer: * <input type="text" value="Male"/>
Number of Products * <input type="text" value="3"/>	
Do the Customer have Credit Card? (1=Yes, 0=No) * <input type="text" value="0"/>	

Predict whether the Customer will leave the bank or not?

The Customer will not leave the bank

Developed by SADISH P

Click **EDS** we get the Code part of these project

Customer Churn Prediction

In this project, we will predict whether a customer will leave the bank or not based on many factors

Following Factors are:

1. Credit score
2. Location of the Customer
3. Gender
4. Age
5. Tenure
6. Account Balance
7. Number of Bank Products Customer Uses
8. Has Credit Card
9. Is Active Member
10. Estimated Salary

```
In [68]: from IPython.display import Image
Image(url="https://miro.medium.com/max/844/1*MyKDLRda6yHGR_8kgVvckg.png")

Out[68]: 

In [69]: # Importing the essential Libraries
import pandas as pd
import numpy as np

In [71]: # Reading the Dataset
df = pd.read_csv('D:/project/Customer_Churn-Deployment-master/churn_Modelling.csv')

In [72]: import tensorflow as tf
```

Conclusion

Customer churn analysis has become a major concern in almost every industry that offers products and services. The model developed will help banks identify clients who are likely to be churners and develop appropriate marketing actions to retain their valuable clients. And this model also supports information about similar customer group to consider which marketing reactions are to be provided. Thus, due to existing customers are retained, it will provide banks with increased profits and revenues.

On the other hand, if the dataset is large and contains many independent variables, ANN - **Artificial Neural Network** is the most computationally effective method that also yields the best results, as discussed previously.

A large dataset may yield similar results for both Univariate Outlier and Multivariate Outlier, as documented.

In this study, the Univariate Outlier and Multivariate Outlier yields different results. Among these two-outlier detection technique, the technique that provided better accuracy, sensitivity and specificity are considered.

However, the aim of this study is to evaluate these models for bank customer churn, and since banks have access to very large datasets, **Univariate Outlier is preferable as this has computational benefits and is more time efficient.**

This concludes that the Customer in France and Germany is more churned than the customers in Spain. Based on Gender Feature, Female are more likely to churn than Male. Credit Card Holders are comparatively churned higher than the other Customers. Non-Activate Customers are Churned more than Others. Customers between the age group of 41 and 60 are churned more than others.

If the Client are Giving some discount or offers to Customer as, if the Credit Card Holder are Churning, Bank Service can give some Special Discount for particular Credit Card Holder. Bank Managers can develop an appropriate marketing action to retain their valuable clients.

Here, we conclude that the Statistical Model, **Artificial Neural Network** gives the high level of accuracy which is 85% with sensitivity and specificity of 86% and 76% respectively using Mahalanobis distance measure.

References

- [1] B. He, Y. Shi, Q. Wan, X. Zhao, Prediction of customer attrition of commercial banks based on SVM model, "2nd International Conference on Information Technology and Quantitative Management, ITQM (2014)".
- [2] A. P. Patil et.al, "Customer Churn Prediction for Retail Business", "(ICECDS-2017)".
- [3] Fa-Gui LIU et.al. "Using Combined Model Approach for Churn Prediction in Telecommunication (FKP-SVM)", "3rd Annual International Conference on Electronics, Electrical Engineering and Information Science (EEEIS 2017)".
- [4] A. O. Oyeniyi & A.B. Adeyemo, Customer Churn Analysis In Banking Sector Using Data Mining, "African Journal of Computing & ICT (2015)".
- [5] F. Abdi and S. Abolmakarem, "Customer Behavior Mining Framework (CBMF) using clustering and classification techniques", "Journal of Industrial Engineering International (2018)".
- [6] H. Sayed, Manal A. Abdel-Fattah, S. Kholief, Predicting Potential Banking Customer Churn using Apache Spark ML and MLlib Packages: a Comparative Study, " International Journal of Advanced Computer Science and Applications, (IJACSA), Vol. 9, No. 11, (2018)".
- [7] A. B. Zoric, Predicting Customer Churn in Banking Industry using Neural Network, " Interdisciplinary Description of Complex Systems 14(2), 116-124, (2016)".
- [8] W. Etaiwi, M. Biltawi and G. Naymat, Evaluation of classification algorithms for banking customer's behaviour under Apache Spark Data Processing System, " The 4th International Symposium on Emerging Information, Communication and Networks, (EICN 2017)"
- [9] S. Singhal and V. Passricha, K-means based SVM for Prediction Analysis, " International Journal of research in electronics and computer engineering, (IJRECE), Vol. 6 issue 3 (2018)"
- [10] The Economist, 2019. A Whole New World: How technology is driving the evolution of intelligent banking, London: The Economist Intelligence Unit (EIU)

- [11] De Caigny, A., Coussement, K. & De Bock, K. W., 2018. A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *European Journal of Operational Research*
- [12] Hair, J. F., Black, J. W. C., Babin, B. J. & Anderson, R. E., 2014. *Multivariate Data Analysis*. 7th ed. Harlow: Pearson international edn.
- [13] McHugh, M. L., 2012. Interrater reliability: the Kappa Statistic. *Biochemia Medica*, 22(3), pp. 276-282.
- [14] James, G., Witten, D., Hastie, T. & Tibshirani, R., 2013. *An Introduction to Statistical Learning: with Applications in R*. New York, NY: Springer New York.
- [15] Colgate, M., Stewart, K. & Kinsella, R., 1996. Customer Defection: A study of the student market in Ireland. *International Journal of Bank Marketing*, 14(3), pp. 23-29.
- [16] Hastie, T., Tibshirani, R. & Friedman, J., 2009. *The Elements of Statistical Learning: data mining, inference, and prediction*. 2nd ed. New York, NY: Springer New York.
- [17] Zhou, J., Yan, J., Yang, L., Wang, M., and Xia, P., “Customer churn prediction model based on LSTM and CNN in music streaming”, *DES tech Transactions on Engineering and Technology Research, (AEMCE)*, (2019).
- [18] Germann, F., Lilien, G., Moorman, and C., Fiedler, L., “Driving customer analytics from the top. Customer Needs and Solutions”, *Customer Needs and Solutions*, 7(3):43–61, (2021).
- [19] Mishra, A. and Reddy, U. S., “A novel approach for churn prediction using deep learning”, *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Coimbatore India, 1–4, (2017).
- [20] Umayaparvathi, V. and Iyakutti, K., “Automated feature selection and churn prediction using deep learning models”, *International Research Journal of Engineering and Technology (IRJET)*, 4(3): 1846–1854, (2017).

Appendix

```
1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv('D:/project/Customer_Churn-Deployment-master/Churn_Modelling.csv')
4 print(df['Geography'].unique())
5 df.isnull().sum()
6 final_dataset = pd.get_dummies(final_dataset, drop_first=True)
7 X=data_m.iloc[:,3:13]
8 Y=data_m["Exited"]
9 X['Geography'].unique()
10 X['Gender'].unique()
11 geography=pd.get_dummies(X['Geography'],drop_first=True)
12 gender=pd.get_dummies(X['Gender'],drop_first=True)
13 X=X.drop(['Geography','Gender'],axis=1)
14 X=pd.concat([X,geography,gender],axis=1)
15 from sklearn.model_selection import train_test_split
16 X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
17 from sklearn.preprocessing import StandardScaler
18 sc =StandardScaler()
19 X_train=sc.fit_transform(X_train)
20 X_test=sc.transform(X_test)
21 import tensorflow as tf
22 print(tf.__version__)
23 from tensorflow.keras.models import Sequential
24 from tensorflow.keras.layers import Dense
25 from tensorflow.keras.layers import LeakyReLU,PreLU,ELU,ReLU
```

;

```
26 classifier=Sequential()
27 classifier.add(Dense(10,input_shape=(11,),activation='relu'))
28 classifier.add(Dense(units=10,activation='relu'))
29 classifier.add(Dense(units=10,activation='relu'))
30 classifier.add(Dense(1,activation='sigmoid'))
31 classifier.summary()
32 import tensorflow
33 opt=tensorflow.keras.optimizers.Adam(learning_rate=0.01)
34 classifier.compile(optimizer=opt,loss='binary_crossentropy',metrics=['accuracy'])
35 import tensorflow as tf
36 model_history=classifier.fit(X_train,y_train,validation_data=(X_test,y_test),batch_size=10,epochs=100)
37 y_pred = classifier.predict(X_test)
38 y_pred = (y_pred >= 0.5)
39 from sklearn.metrics import confusion_matrix, classification_report
40 cm=confusion_matrix(y_test,y_pred)
41 print(cm)
42
43 print("classification_report")
44 print(classification_report(y_pred, y_test, digits=2))
45 from sklearn.metrics import accuracy_score
46 score=accuracy_score(y_pred,y_test)
47 score
48 classifier.save("model1.h5")
```

Done by

Sadish P

7299279600

apjsadish2706@gmail.com

github= <https://github.com/sadishpitchandi>

Linkedin: <https://www.linkedin.com/in/sadish-p-a29745217/>