|  |  |
| --- | --- |
| Algorithm: Artificial Neural Network (ANN) | |
| USN : 1MS17CS143 | NAME : Sathvik K P |
| USN : 1MS17CS148 | NAME : Sathvik B |

**Description of the Algorithm:**

An artificial neuron network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output.

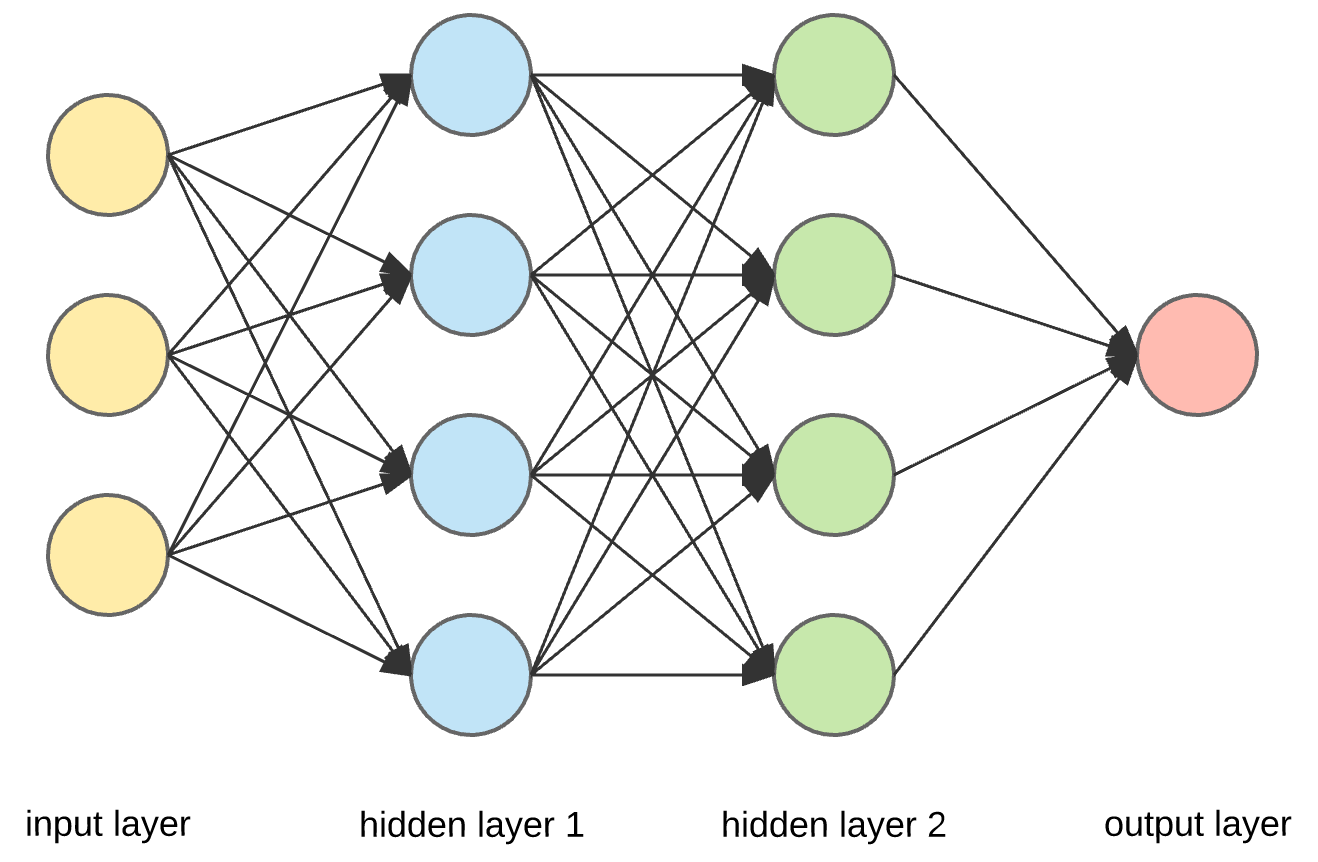
ANNs are considered nonlinear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found.

ANN is also known as a neural network.

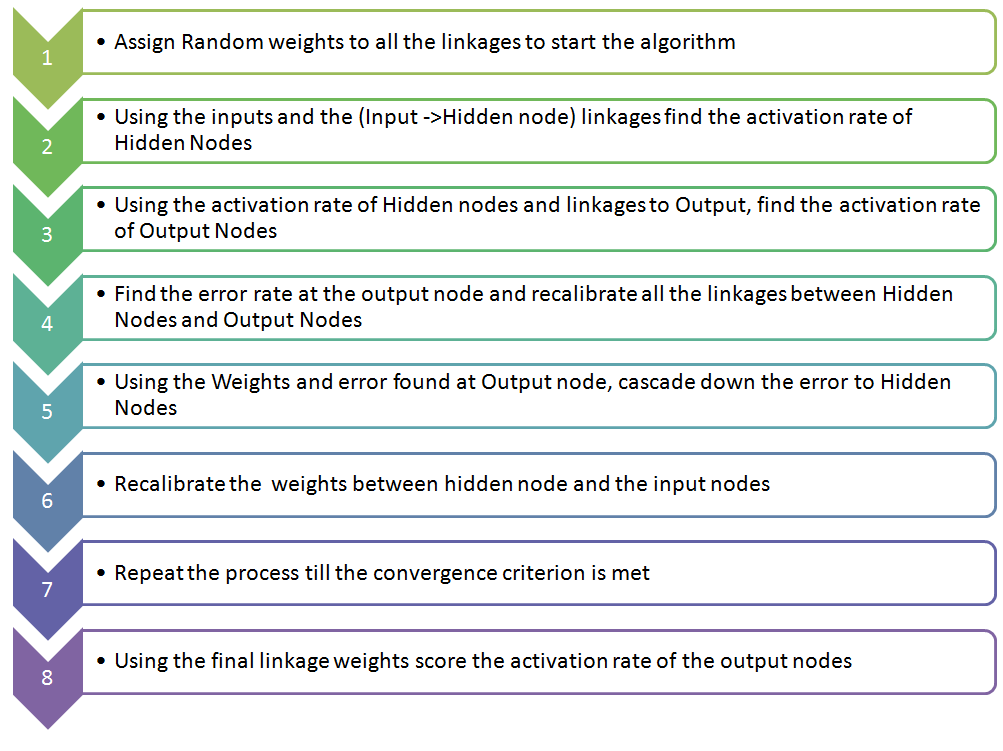
An ANN has several advantages but one of the most recognized of these is the fact that it can actually learn from observing data sets. In this way, ANN is used as a random function approximation tool. These types of tools help estimate the most cost-effective and ideal methods for arriving at solutions while defining computing functions or distributions. ANN takes data samples rather than entire data sets to arrive at solutions, which saves both time and money. ANNs are considered fairly simple mathematical models to enhance existing data analysis technologies.

ANNs have three layers that are interconnected. The first layer consists of input neurons. Those neurons send data on to the second layer, which in turn sends the output neurons to the third layer.

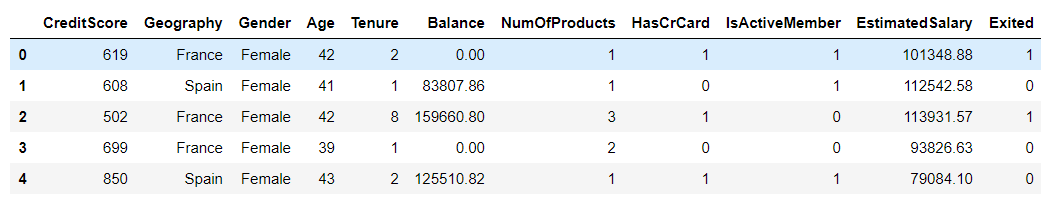
Training an artificial neural network involves choosing from allowed models for which there are several associated algorithms.



**Algorithm Pseudocode:**



**Data set Used: (Attach Screen shot of the few rows)**



**Challenges faced during the implementation of the program:**

1. Conversion of categorical attributes to numerical as ANN cannot work with categorical data.
2. Poor accuracy on un-processed data
3. On normalization, accuracy greatly improved
4. Time for epochs increased with increased batch count
5. Finding optimal activation function was difficult

**Code:**

import numpy as np

import pandas as pd

from random import randint

from scipy import stats

from copy import deepcopy

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.decomposition import PCA

from sklearn.feature\_selection import chi2, SelectKBest

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix,roc\_auc\_score,average\_precision\_score

import keras as kr

from keras.models import Sequential

from keras.layers import Dense

df = pd.read\_csv('Churn\_Modelling.csv')

df.isnull().sum()

df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)

df['Geography'] = df['Geography'].apply(lambda x: 0 if x=='France' else (1 if x=='Spain' else 2))

df['Gender'] = df['Gender'].apply(lambda x: 0 if x=='Female' else 1)

corr\_matrix = df.corr()

scaler = StandardScaler()

df[['CreditScore']] = scaler.fit\_transform(df[['CreditScore']])

df[['Geography']] = scaler.fit\_transform(df[['Geography']])

df[['Gender']] = scaler.fit\_transform(df[['Gender']])

df[['Age']] = scaler.fit\_transform(df[['Age']])

df[['Tenure']] = scaler.fit\_transform(df[['Tenure']])

df[['Balance']] = scaler.fit\_transform(df[['Balance']])

df[['NumOfProducts']] = scaler.fit\_transform(df[['NumOfProducts']])

df[['HasCrCard']] = scaler.fit\_transform(df[['HasCrCard']])

df[['IsActiveMember']] = scaler.fit\_transform(df[['IsActiveMember']])

df[['EstimatedSalary']] = scaler.fit\_transform(df[['EstimatedSalary']])

X = df[['CreditScore','Geography','Gender', 'Age', 'Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary']]

Y=df[['Exited']]

X\_train, X\_test, Y\_train, Y\_test = traimodel = Sequential()

model.add(Dense(15, input\_dim=10, activation='tanh'))

model.add(Dense(15, input\_dim=15, activation='tanh'))

model.add(Dense(1, activation='tanh'))

model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

history = model.fit(X\_train, Y\_train, epochs=10, batch\_size=10)n\_test\_split(X, Y, test\_size=0.2, random\_state=42)

Y\_pred = model.predict(X\_test)

Y\_pred = (Y\_pred >= 0.5)

cm = confusion\_matrix(Y\_test, Y\_pred)

sns.heatmap(cm,annot=True)

print("Accuracy : ",(cm[0][0]+cm[1][1])/(cm[0][0]+cm[1][1]+cm[1][0]+cm[0][1]))

print("Precision : ",(cm[0][0])/(cm[0][0]+cm[0][1]))

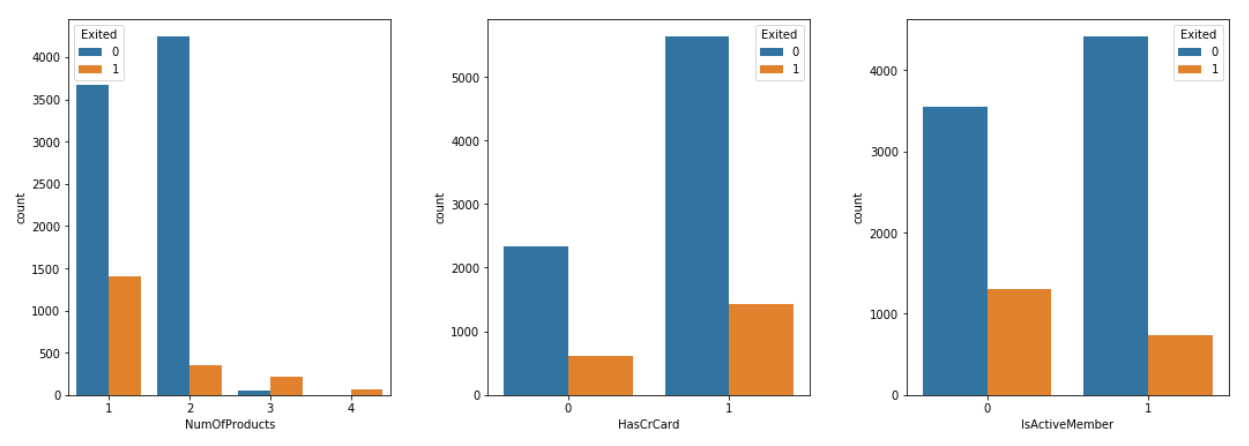
print("Recall : ",(cm[0][0])/(cm[0][0]+cm[1][0]))

ppp=(cm[0][0])/(cm[0][0]+cm[0][1]);rrr=(cm[0][0])/(cm[0][0]+cm[1][0]);fff=2\*ppp\*rrr/(ppp+rrr)

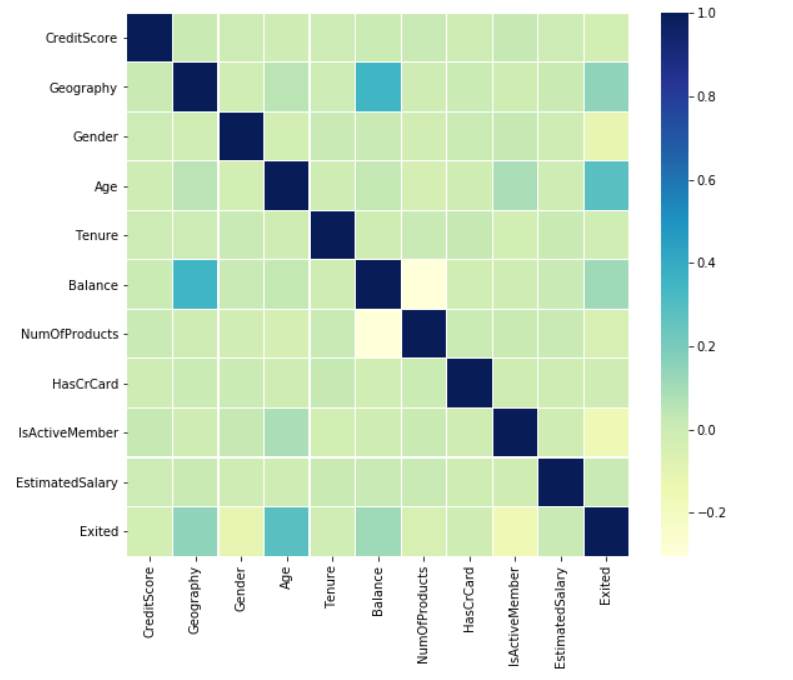
print("F1 score : ",fff)

**Output: (Screen shots)**

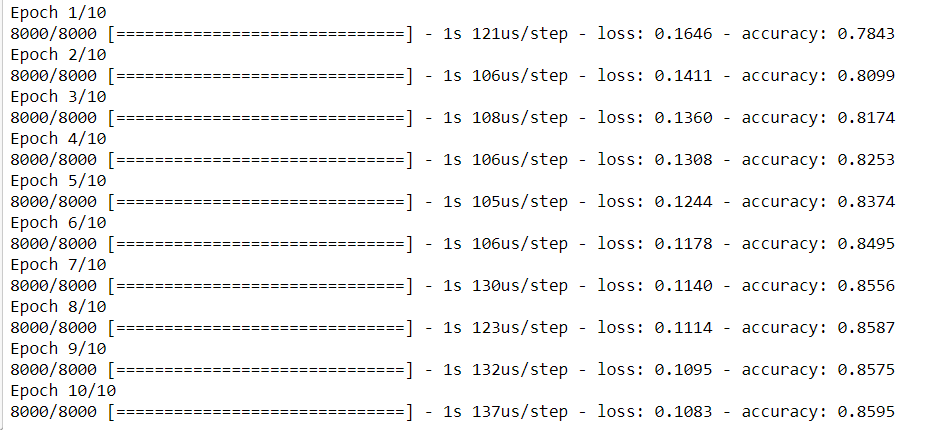
EDA



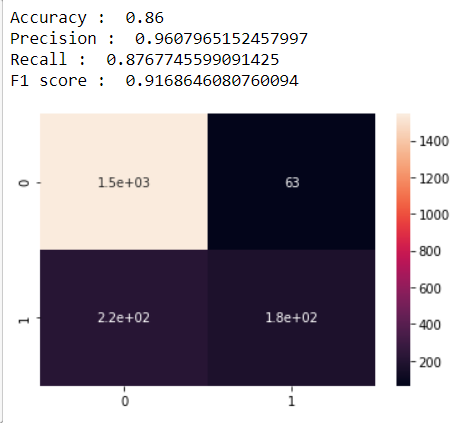
Correlation matrix



Epochs



Confusion matrix and results



**References:**

1. <https://www.analyticsvidhya.com/blog/2014/10/ann-work-simplified/>
2. <https://en.wikipedia.org/wiki/Artificial_neural_network>
3. <https://www.techopedia.com/definition/5967/artificial-neural-network-ann>