DS640 Assignment 5

October 22, 2018

1 Manav Patel

1.1 Introduction

Repeat customers are the most valuable customers you have.

Repeat customers:

- Are loyal and satisfied
- Make multiple purchases
- Talk about your brand
- Refer your brand to their friends

However, maintaining your relationship with loyal customers can be tricky.These relationships need to be maintained with a consistent and rewarding customer experience. When your customers are happy, your business will prosper.

What is Churn ? Customers that have churned are customers that have cut ties with a business or brand.

What is Churn Rate used for ? Churn rate is a measurement of the health of your business. Churn measures:

- The success of increasing customer retention for a company
- The changes that affect customer retention
- Customer Lifetime Value
- The type of customers that have the highest retention rate with your company
- Much more!

Today, we are going to analyse a customer dataset of a bank to provide them with valuable insights for their ongoing customer retention issue. We would apply the ANN(Artificial Neural Networks) to classify the dataset and provide the bank with the best fit model. The problem statement is mentioned in detail below.

1.2 Problem Statement

The bank has been seen unusual churn rates for their customers (churn is when people leave the company). They want to understand the problem for this unusual high churn rates. Here we have been provided with a sample of their, last six months customers whose charecteristics like

- credit score

- geography
- gender
- age
- tenure (how many years a customer is with the bank)
- balance
- number of products a customer had with the bank
- if he/she had a credit card
- if he/she is an active member
- estimated salary (the bank has estimated the salary based of the data they had)
- exited (if a customer has left the bank or not within the last six months)

Our task is to create a classification model for predicting customers at risk of churning. From this classification model the bank could gain useful insights and can take relevant actions to prevent further unusual churn rate and improve the overall customer experience.

1.3 Data Preprossesing

In [89]:
```
# Importing the required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [90]:
```
#Reading the dataset
dataset = pd.read_csv('/Users/Lenovo/Desktop/Churn_Modelling.csv')
dataset
```

Out[90]:
```
    RowNumber  CustomerId    Surname  CreditScore Geography  Gender  \
0           1    15634602   Hargrave          619    France  Female
1           2    15647311       Hill          608     Spain  Female
2           3    15619304       Onio          502    France  Female
3           4    15701354       Boni          699    France  Female
4           5    15737888   Mitchell          850     Spain  Female
5           6    15574012        Chu          645     Spain    Male
6           7    15592531   Bartlett          822    France    Male
7           8    15656148     Obinna          376   Germany  Female
8           9    15792365         He          501    France    Male
9          10    15592389         H?          684    France    Male
10         11    15767821     Bearce          528    France    Male
11         12    15737173    Andrews          497     Spain    Male
12         13    15632264        Kay          476    France  Female
13         14    15691483       Chin          549    France  Female
14         15    15600882      Scott          635     Spain  Female
15         16    15643966    Goforth          616   Germany    Male
16         17    15737452      Romeo          653   Germany    Male
17         18    15788218  Henderson          549     Spain  Female
18         19    15661507    Muldrow          587     Spain    Male
19         20    15568982        Hao          726    France  Female
20         21    15577657   McDonald          732    France    Male
21         22    15597945   Dellucci          636     Spain  Female
```

22	23	15699309	Gerasimov	510	Spain	Female
23	24	15725737	Mosman	669	France	Male
24	25	15625047	Yen	846	France	Female
25	26	15738191	Maclean	577	France	Male
26	27	15736816	Young	756	Germany	Male
27	28	15700772	Nebechi	571	France	Male
28	29	15728693	McWilliams	574	Germany	Female
29	30	15656300	Lucciano	411	France	Male
...	...	...	...	...	...	...
9970	9971	15587133	Thompson	518	France	Male
9971	9972	15721377	Chou	833	France	Female
9972	9973	15747927	Ch'in	758	France	Male
9973	9974	15806455	Miller	611	France	Male
9974	9975	15695474	Barker	583	France	Male
9975	9976	15666295	Smith	610	Germany	Male
9976	9977	15656062	Azikiwe	637	France	Female
9977	9978	15579969	Mancini	683	France	Female
9978	9979	15703563	P'eng	774	France	Male
9979	9980	15692664	Diribe	677	France	Female
9980	9981	15719276	T'ao	741	Spain	Male
9981	9982	15672754	Burbidge	498	Germany	Male
9982	9983	15768163	Griffin	655	Germany	Female
9983	9984	15656710	Cocci	613	France	Male
9984	9985	15696175	Echezonachukwu	602	Germany	Male
9985	9986	15586914	Nepean	659	France	Male
9986	9987	15581736	Bartlett	673	Germany	Male
9987	9988	15588839	Mancini	606	Spain	Male
9988	9989	15589329	Pirozzi	775	France	Male
9989	9990	15605622	McMillan	841	Spain	Male
9990	9991	15798964	Nkemakonam	714	Germany	Male
9991	9992	15769959	Ajuluchukwu	597	France	Female
9992	9993	15657105	Chukwualuka	726	Spain	Male
9993	9994	15569266	Rahman	644	France	Male
9994	9995	15719294	Wood	800	France	Female
9995	9996	15606229	Obijiaku	771	France	Male
9996	9997	15569892	Johnstone	516	France	Male
9997	9998	15584532	Liu	709	France	Female
9998	9999	15682355	Sabbatini	772	Germany	Male
9999	10000	15628319	Walker	792	France	Female

	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	42	2	0.00	1	1	1	
1	41	1	83807.86	1	0	1	
2	42	8	159660.80	3	1	0	
3	39	1	0.00	2	0	0	
4	43	2	125510.82	1	1	1	
5	44	8	113755.78	2	1	0	
6	50	7	0.00	2	1	1	

7	29	4	115046.74	4	1	0
8	44	4	142051.07	2	0	1
9	27	2	134603.88	1	1	1
10	31	6	102016.72	2	0	0
11	24	3	0.00	2	1	0
12	34	10	0.00	2	1	0
13	25	5	0.00	2	0	0
14	35	7	0.00	2	1	1
15	45	3	143129.41	2	0	1
16	58	1	132602.88	1	1	0
17	24	9	0.00	2	1	1
18	45	6	0.00	1	0	0
19	24	6	0.00	2	1	1
20	41	8	0.00	2	1	1
21	32	8	0.00	2	1	0
22	38	4	0.00	1	1	0
23	46	3	0.00	2	0	1
24	38	5	0.00	1	1	1
25	25	3	0.00	2	0	1
26	36	2	136815.64	1	1	1
27	44	9	0.00	2	0	0
28	43	3	141349.43	1	1	1
29	29	0	59697.17	2	1	1
...	...	...	...	...	...	...
9970	42	7	151027.05	2	1	0
9971	34	3	144751.81	1	0	0
9972	26	4	155739.76	1	1	0
9973	27	7	0.00	2	1	1
9974	33	7	122531.86	1	1	0
9975	50	1	113957.01	2	1	0
9976	33	7	103377.81	1	1	0
9977	32	9	0.00	2	1	1
9978	40	9	93017.47	2	1	0
9979	58	1	90022.85	1	0	1
9980	35	6	74371.49	1	0	0
9981	42	3	152039.70	1	1	1
9982	46	7	137145.12	1	1	0
9983	40	4	0.00	1	0	0
9984	35	7	90602.42	2	1	1
9985	36	6	123841.49	2	1	0
9986	47	1	183579.54	2	0	1
9987	30	8	180307.73	2	1	1
9988	30	4	0.00	2	1	0
9989	28	4	0.00	2	1	1
9990	33	3	35016.60	1	1	0
9991	53	4	88381.21	1	1	0
9992	36	2	0.00	1	1	0
9993	28	7	155060.41	1	1	0

9994	29	2	0.00	2	0	0
9995	39	5	0.00	2	1	0
9996	35	10	57369.61	1	1	1
9997	36	7	0.00	1	0	1
9998	42	3	75075.31	2	1	0
9999	28	4	130142.79	1	1	0

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
5	149756.71	1
6	10062.80	0
7	119346.88	1
8	74940.50	0
9	71725.73	0
10	80181.12	0
11	76390.01	0
12	26260.98	0
13	190857.79	0
14	65951.65	0
15	64327.26	0
16	5097.67	1
17	14406.41	0
18	158684.81	0
19	54724.03	0
20	170886.17	0
21	138555.46	0
22	118913.53	1
23	8487.75	0
24	187616.16	0
25	124508.29	0
26	170041.95	0
27	38433.35	0
28	100187.43	0
29	53483.21	0
...	...	...
9970	119377.36	0
9971	166472.81	0
9972	171552.02	0
9973	157474.10	0
9974	13549.24	0
9975	196526.55	1
9976	84419.78	0
9977	24991.92	0
9978	191608.97	0

```
9979       2988.28       0
9980      99595.67       0
9981      53445.17       1
9982     115146.40       1
9983     151325.24       0
9984      51695.41       0
9985      96833.00       0
9986      34047.54       0
9987       1914.41       0
9988      49337.84       0
9989     179436.60       0
9990      53667.08       0
9991      69384.71       1
9992     195192.40       0
9993      29179.52       0
9994     167773.55       0
9995      96270.64       0
9996     101699.77       0
9997      42085.58       1
9998      92888.52       1
9999      38190.78       0

[10000 rows x 14 columns]
```

In [91]:
```
#Summary of the dataset
dataset.describe()
```

Out[91]:
```
         RowNumber    CustomerId   CreditScore           Age        Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000  10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800      5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806      2.892174
min        1.00000  1.556570e+07    350.000000     18.000000      0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000      3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000      5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000      7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000     10.000000

             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
mean    76485.889288       1.530200      0.70550        0.515100
std     62397.405202       0.581654      0.45584        0.499797
min         0.000000       1.000000      0.00000        0.000000
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
```

```
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

We can see that their is no missing value in the dataset as the size of the dataset is 10000x14 and all the attributes have a count of 10000.

We do not consider the row number, the customer id and the surname as independent variables as they would have no impact on our dependent variable. Hence we trim the dataset and divide the dataset as follows:

```
In [92]: X = dataset.iloc[:, 3:13].values #Feature Variable
         y = dataset.iloc[:, 13].values #Target Variable

In [93]: print(X)
         print(y)

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

Our model needs in input numerical data, so, we need to encode categorical data into numerical data. In This case we have Geography (France, Spain and Germany) and Gender (Male and Female). For Geography we will have 0,1,2 instead of France, Spain and Gemany and 0,1 instead of Gender.

```
In [94]: #Converting to Variables
         from sklearn.preprocessing import LabelEncoder, OneHotEncoder
         labelencoder_X_1 = LabelEncoder()
         X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
         labelencoder_X_2 = LabelEncoder()
         X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
```

We have a binary value for the column 'Gender' hence we don't have issues with the encoding.

But, for the column 'Geography', we have 3 possible values and considering that our categorical values have no comparison (for example, Germany is not higher than France), we need to create dummy variables, so we will create 3 columns, one for each country where we will have 1 if the customer is from that country and 0 if not. And to avoid the dummy variable trap, we will also delete 1 of the 3 columns:

```
In [95]: #Creating a dummy variable
         onehotencoder = OneHotEncoder(categorical_features = [1])
         X = onehotencoder.fit_transform(X).toarray()
         X = X[:, 1:]
```

We now split the dataset into training and testing sets.

```
In [96]: #Splitting the dataset into Training and testing sets
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_sta
```

We need to apply feature scaling to the dataset to standardize the dataset to convert them into same sclae for faster computation of results.

```
In [97]: #Feature scaling
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train = sc.fit_transform(X_train)
         X_test = sc.transform(X_test)
```

1.4 Artificial Neural Network Clustering

```
In [98]: #Importing the required libraries
         import keras
         from keras.models import Sequential
         from keras.layers import Dense
```

1.4.1 Model 1

We now create the ANN with one hidden layer and having 6 nodes (Half of feature and target variables) with activation function of the hidden and the input layer as 'relu' (rectified linear units) and the output layer 'sigmoid' as we need binary result for our classification model. Then we compile the model with SGD 'adam' as its the best which can be used in thos case.

```
In [99]: #Creating a Neural network
         classifier = Sequential()
         classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim
         classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))#The hidd
         classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))#The O
         classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accu

C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: Update your
  after removing the cwd from sys.path.
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: UserWarning: Update your
  """
```

```
In [100]: #Fitting the Neural network model on the testing data
          classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

```
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning: The `nb_epoch
```

```
Epoch 1/100
8000/8000 [==============================] - 4s 499us/step - loss: 0.4931 - acc: 0.7954
Epoch 2/100
8000/8000 [==============================] - 2s 265us/step - loss: 0.4302 - acc: 0.7960
Epoch 3/100
8000/8000 [==============================] - 2s 255us/step - loss: 0.4255 - acc: 0.7960
Epoch 4/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.4210 - acc: 0.7999
Epoch 5/100
8000/8000 [==============================] - 2s 233us/step - loss: 0.4182 - acc: 0.8226
Epoch 6/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.4161 - acc: 0.8262
Epoch 7/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.4144 - acc: 0.8294
Epoch 8/100
8000/8000 [==============================] - 1s 184us/step - loss: 0.4132 - acc: 0.8292
Epoch 9/100
8000/8000 [==============================] - 1s 178us/step - loss: 0.4125 - acc: 0.8312
Epoch 10/100
8000/8000 [==============================] - 2s 200us/step - loss: 0.4113 - acc: 0.8317
Epoch 11/100
8000/8000 [==============================] - 2s 280us/step - loss: 0.4103 - acc: 0.8336
Epoch 12/100
8000/8000 [==============================] - 2s 245us/step - loss: 0.4094 - acc: 0.8326
Epoch 13/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.4088 - acc: 0.8324
Epoch 14/100
8000/8000 [==============================] - 2s 227us/step - loss: 0.4086 - acc: 0.8350
Epoch 15/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.4079 - acc: 0.8349
Epoch 16/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.4074 - acc: 0.8335
Epoch 17/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.4068 - acc: 0.8341
Epoch 18/100
8000/8000 [==============================] - 2s 216us/step - loss: 0.4065 - acc: 0.8340
Epoch 19/100
8000/8000 [==============================] - 2s 273us/step - loss: 0.4059 - acc: 0.8341
Epoch 20/100
8000/8000 [==============================] - 2s 285us/step - loss: 0.4056 - acc: 0.8346
Epoch 21/100
```

```
8000/8000 [==============================] - 2s 256us/step - loss: 0.4056 - acc: 0.8347
Epoch 22/100
8000/8000 [==============================] - 2s 210us/step - loss: 0.4049 - acc: 0.8350
Epoch 23/100
8000/8000 [==============================] - 1s 186us/step - loss: 0.4047 - acc: 0.8325
Epoch 24/100
8000/8000 [==============================] - 1s 171us/step - loss: 0.4045 - acc: 0.8350
Epoch 25/100
8000/8000 [==============================] - 1s 173us/step - loss: 0.4039 - acc: 0.8336
Epoch 26/100
8000/8000 [==============================] - 1s 165us/step - loss: 0.4038 - acc: 0.8365
Epoch 27/100
8000/8000 [==============================] - 1s 168us/step - loss: 0.4038 - acc: 0.8352
Epoch 28/100
8000/8000 [==============================] - 2s 193us/step - loss: 0.4033 - acc: 0.8342
Epoch 29/100
8000/8000 [==============================] - 2s 271us/step - loss: 0.4038 - acc: 0.8341
Epoch 30/100
8000/8000 [==============================] - 2s 261us/step - loss: 0.4034 - acc: 0.8345
Epoch 31/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.4031 - acc: 0.8354
Epoch 32/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.4027 - acc: 0.8346
Epoch 33/100
8000/8000 [==============================] - 2s 230us/step - loss: 0.4028 - acc: 0.8371
Epoch 34/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.4024 - acc: 0.8341
Epoch 35/100
8000/8000 [==============================] - 2s 213us/step - loss: 0.4019 - acc: 0.8336
Epoch 36/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.4019 - acc: 0.8347
Epoch 37/100
8000/8000 [==============================] - 2s 239us/step - loss: 0.4022 - acc: 0.8360
Epoch 38/100
8000/8000 [==============================] - 2s 208us/step - loss: 0.4019 - acc: 0.8345
Epoch 39/100
8000/8000 [==============================] - 2s 193us/step - loss: 0.4020 - acc: 0.8372
Epoch 40/100
8000/8000 [==============================] - 1s 169us/step - loss: 0.4025 - acc: 0.8351
Epoch 41/100
8000/8000 [==============================] - 2s 191us/step - loss: 0.4018 - acc: 0.8361
Epoch 42/100
8000/8000 [==============================] - 1s 169us/step - loss: 0.4015 - acc: 0.8337
Epoch 43/100
8000/8000 [==============================] - 2s 192us/step - loss: 0.4019 - acc: 0.8360
Epoch 44/100
8000/8000 [==============================] - 2s 255us/step - loss: 0.4017 - acc: 0.8344
Epoch 45/100
```

```
8000/8000 [==============================] - 2s 245us/step - loss: 0.4015 - acc: 0.8350
Epoch 46/100
8000/8000 [==============================] - 2s 223us/step - loss: 0.4017 - acc: 0.8334
Epoch 47/100
8000/8000 [==============================] - 2s 254us/step - loss: 0.4017 - acc: 0.8349
Epoch 48/100
8000/8000 [==============================] - 2s 250us/step - loss: 0.4012 - acc: 0.8362
Epoch 49/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.4015 - acc: 0.8365
Epoch 50/100
8000/8000 [==============================] - 2s 258us/step - loss: 0.4013 - acc: 0.8345
Epoch 51/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.4016 - acc: 0.8349
Epoch 52/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.4012 - acc: 0.8360
Epoch 53/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.4009 - acc: 0.8354
Epoch 54/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.4011 - acc: 0.8346
Epoch 55/100
8000/8000 [==============================] - 2s 254us/step - loss: 0.4011 - acc: 0.8342
Epoch 56/100
8000/8000 [==============================] - 2s 250us/step - loss: 0.4009 - acc: 0.8351
Epoch 57/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.4011 - acc: 0.8342
Epoch 58/100
8000/8000 [==============================] - 2s 219us/step - loss: 0.4005 - acc: 0.8350
Epoch 59/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.4004 - acc: 0.8339
Epoch 60/100
8000/8000 [==============================] - 2s 214us/step - loss: 0.4008 - acc: 0.8330
Epoch 61/100
8000/8000 [==============================] - 2s 214us/step - loss: 0.4010 - acc: 0.8346
Epoch 62/100
8000/8000 [==============================] - 2s 217us/step - loss: 0.4007 - acc: 0.8347
Epoch 63/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.4005 - acc: 0.8346
Epoch 64/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.4007 - acc: 0.8354
Epoch 65/100
8000/8000 [==============================] - 2s 209us/step - loss: 0.4008 - acc: 0.8344
Epoch 66/100
8000/8000 [==============================] - 2s 195us/step - loss: 0.4010 - acc: 0.8345
Epoch 67/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.4003 - acc: 0.8349
Epoch 68/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.4003 - acc: 0.8342
Epoch 69/100
```

```
8000/8000 [==============================] - 2s 224us/step - loss: 0.4005 - acc: 0.8359
Epoch 70/100
8000/8000 [==============================] - 2s 219us/step - loss: 0.4006 - acc: 0.8346
Epoch 71/100
8000/8000 [==============================] - 2s 212us/step - loss: 0.4001 - acc: 0.8349
Epoch 72/100
8000/8000 [==============================] - 2s 219us/step - loss: 0.4003 - acc: 0.8347
Epoch 73/100
8000/8000 [==============================] - 2s 256us/step - loss: 0.4005 - acc: 0.8359
Epoch 74/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.4004 - acc: 0.8340
Epoch 75/100
8000/8000 [==============================] - 2s 242us/step - loss: 0.4007 - acc: 0.8342
Epoch 76/100
8000/8000 [==============================] - 2s 239us/step - loss: 0.4005 - acc: 0.8361
Epoch 77/100
8000/8000 [==============================] - 2s 198us/step - loss: 0.3999 - acc: 0.8357
Epoch 78/100
8000/8000 [==============================] - 2s 200us/step - loss: 0.4004 - acc: 0.8345 1s - lo
Epoch 79/100
8000/8000 [==============================] - 1s 166us/step - loss: 0.4000 - acc: 0.8342
Epoch 80/100
8000/8000 [==============================] - 1s 173us/step - loss: 0.4003 - acc: 0.8364
Epoch 81/100
8000/8000 [==============================] - 2s 193us/step - loss: 0.4005 - acc: 0.8345
Epoch 82/100
8000/8000 [==============================] - 2s 199us/step - loss: 0.4003 - acc: 0.8362
Epoch 83/100
8000/8000 [==============================] - 2s 218us/step - loss: 0.4003 - acc: 0.8349
Epoch 84/100
8000/8000 [==============================] - 2s 201us/step - loss: 0.4002 - acc: 0.8355
Epoch 85/100
8000/8000 [==============================] - 1s 178us/step - loss: 0.3999 - acc: 0.8346
Epoch 86/100
8000/8000 [==============================] - 1s 179us/step - loss: 0.4000 - acc: 0.8341
Epoch 87/100
8000/8000 [==============================] - 2s 196us/step - loss: 0.3997 - acc: 0.8364 0s - lo
Epoch 88/100
8000/8000 [==============================] - 2s 239us/step - loss: 0.4002 - acc: 0.8355
Epoch 89/100
8000/8000 [==============================] - 2s 213us/step - loss: 0.3996 - acc: 0.8359
Epoch 90/100
8000/8000 [==============================] - 2s 208us/step - loss: 0.3996 - acc: 0.8342
Epoch 91/100
8000/8000 [==============================] - 2s 209us/step - loss: 0.3996 - acc: 0.8336
Epoch 92/100
8000/8000 [==============================] - 2s 238us/step - loss: 0.3998 - acc: 0.8351
Epoch 93/100
```

```
8000/8000 [==============================] - 2s 248us/step - loss: 0.3999 - acc: 0.8364
Epoch 94/100
8000/8000 [==============================] - 2s 254us/step - loss: 0.3999 - acc: 0.8351
Epoch 95/100
8000/8000 [==============================] - 2s 212us/step - loss: 0.3998 - acc: 0.8365
Epoch 96/100
8000/8000 [==============================] - 2s 212us/step - loss: 0.3997 - acc: 0.8369
Epoch 97/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.3995 - acc: 0.8342
Epoch 98/100
8000/8000 [==============================] - 2s 213us/step - loss: 0.3995 - acc: 0.8359
Epoch 99/100
8000/8000 [==============================] - 2s 204us/step - loss: 0.3992 - acc: 0.8347
Epoch 100/100
8000/8000 [==============================] - 2s 206us/step - loss: 0.3995 - acc: 0.8369
```

```
Out[100]: <keras.callbacks.History at 0x1a082447278>
```

```
In [101]: #Predicting the testing set results
          y_pred = classifier.predict(X_test)
          y_pred = (y_pred > 0.5)
```

```
In [102]: #Creating the confusion matrix
          from sklearn.metrics import confusion_matrix
          cm = confusion_matrix(y_test, y_pred)
          print(cm)
```

```
[[1539   56]
 [ 256  149]]
```

```
In [103]: #Calculating the accuracy of the model
          accuracy_ANN = (cm[0,0]+cm[1,1])/ (cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
          accuracy_ANN
```

```
Out[103]: 0.844
```

What is Specificity ?

Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified. Specificity relates to the test's ability to correctly reject customers who did not churn and actually did'nt churn.

```
In [104]: #Calculating the specificity of the model
          specificity_ANN =(cm[1,1]/(cm[1,1]+cm[0,1]))
          specificity_ANN
```

```
Out[104]: 0.7268292682926829
```

What is Sensitivity ?

Sensitivity (also called the true positive rate) measures the proportion of actual positives that are correctly identified. Sensitivity refers to the test's ability to correctly detect people who are going to churn and actually do churn.

```
In [105]: #Calculating the sensitivity of the model
          sensitivity_ANN =(cm[0,0]/(cm[0,0]+cm[1,0]))
          sensitivity_ANN

Out[105]: 0.8573816155988858

In [125]: #loss function
          classifier_1.evaluate(X_test,y_pred1, verbose=1)

2000/2000 [==============================] - 0s 38us/step


Out[125]: [0.17338521599769594, 1.0]
```

1.4.2 Model 2

We now create the ANN with one hidden layer and having 6 nodes (Half of feature and target variables) with activation function of the hidden and the input layer as 'tanh' (hyperbolic tangent) and the output layer 'sigmoid' as we need binary result for our classification model. Then we compile the model with SGD 'adam' as its the best which can be used in this case.

```
In [106]: #Creating a Neural Network model
          classifier_1 = Sequential()
          classifier_1.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh', input_
          classifier_1.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh'))
          classifier_1.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
          classifier_1.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['a

C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: Update your
  after removing the cwd from sys.path.
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: UserWarning: Update your
  """


In [107]: #Fitting the model on testing data
          classifier_1.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)

C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning: The `nb_epoc
```

```
Epoch 1/100
8000/8000 [==============================] - 3s 344us/step - loss: 0.4769 - acc: 0.8085
Epoch 2/100
8000/8000 [==============================] - 2s 212us/step - loss: 0.4335 - acc: 0.8121
Epoch 3/100
8000/8000 [==============================] - 2s 264us/step - loss: 0.4322 - acc: 0.8102
Epoch 4/100
8000/8000 [==============================] - 2s 258us/step - loss: 0.4293 - acc: 0.8116
Epoch 5/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.4202 - acc: 0.8144
Epoch 6/100
8000/8000 [==============================] - 2s 213us/step - loss: 0.4023 - acc: 0.8306
Epoch 7/100
8000/8000 [==============================] - 2s 217us/step - loss: 0.3820 - acc: 0.8391
Epoch 8/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.3660 - acc: 0.8510
Epoch 9/100
8000/8000 [==============================] - 2s 218us/step - loss: 0.3565 - acc: 0.8545
Epoch 10/100
8000/8000 [==============================] - 2s 242us/step - loss: 0.3520 - acc: 0.8579
Epoch 11/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3489 - acc: 0.8584
Epoch 12/100
8000/8000 [==============================] - 2s 263us/step - loss: 0.3490 - acc: 0.8579
Epoch 13/100
8000/8000 [==============================] - 2s 259us/step - loss: 0.3475 - acc: 0.8599
Epoch 14/100
8000/8000 [==============================] - 2s 230us/step - loss: 0.3476 - acc: 0.8584
Epoch 15/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.3464 - acc: 0.8570
Epoch 16/100
8000/8000 [==============================] - 2s 211us/step - loss: 0.3460 - acc: 0.8584
Epoch 17/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.3458 - acc: 0.8584
Epoch 18/100
8000/8000 [==============================] - 2s 218us/step - loss: 0.3454 - acc: 0.8605
Epoch 19/100
8000/8000 [==============================] - 2s 218us/step - loss: 0.3449 - acc: 0.8604
Epoch 20/100
8000/8000 [==============================] - 2s 246us/step - loss: 0.3449 - acc: 0.8600
Epoch 21/100
8000/8000 [==============================] - 2s 269us/step - loss: 0.3447 - acc: 0.8591
Epoch 22/100
8000/8000 [==============================] - 2s 261us/step - loss: 0.3443 - acc: 0.8565
Epoch 23/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3431 - acc: 0.8580
Epoch 24/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.3432 - acc: 0.8571
```

```
Epoch 25/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.3442 - acc: 0.8586
Epoch 26/100
8000/8000 [==============================] - 2s 212us/step - loss: 0.3427 - acc: 0.8589
Epoch 27/100
8000/8000 [==============================] - 2s 229us/step - loss: 0.3423 - acc: 0.8577
Epoch 28/100
8000/8000 [==============================] - 2s 214us/step - loss: 0.3426 - acc: 0.8579
Epoch 29/100
8000/8000 [==============================] - 2s 284us/step - loss: 0.3425 - acc: 0.8592
Epoch 30/100
8000/8000 [==============================] - 2s 260us/step - loss: 0.3418 - acc: 0.8569
Epoch 31/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3410 - acc: 0.8582
Epoch 32/100
8000/8000 [==============================] - 2s 227us/step - loss: 0.3403 - acc: 0.8587
Epoch 33/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.3399 - acc: 0.8604
Epoch 34/100
8000/8000 [==============================] - 2s 230us/step - loss: 0.3393 - acc: 0.8619
Epoch 35/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.3394 - acc: 0.8599
Epoch 36/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.3387 - acc: 0.8601
Epoch 37/100
8000/8000 [==============================] - 2s 269us/step - loss: 0.3378 - acc: 0.8590
Epoch 38/100
8000/8000 [==============================] - 2s 269us/step - loss: 0.3385 - acc: 0.8607
Epoch 39/100
8000/8000 [==============================] - 2s 274us/step - loss: 0.3370 - acc: 0.8591
Epoch 40/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3369 - acc: 0.8605
Epoch 41/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3372 - acc: 0.8595
Epoch 42/100
8000/8000 [==============================] - 2s 242us/step - loss: 0.3358 - acc: 0.8612
Epoch 43/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3372 - acc: 0.8602
Epoch 44/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.3361 - acc: 0.8622
Epoch 45/100
8000/8000 [==============================] - 2s 258us/step - loss: 0.3365 - acc: 0.8619
Epoch 46/100
8000/8000 [==============================] - 2s 274us/step - loss: 0.3357 - acc: 0.8614
Epoch 47/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3360 - acc: 0.8605
Epoch 48/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.3352 - acc: 0.8605 1s - l
```

```
Epoch 49/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3357 - acc: 0.8614
Epoch 50/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.3350 - acc: 0.8632
Epoch 51/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.3358 - acc: 0.8619
Epoch 52/100
8000/8000 [==============================] - 2s 229us/step - loss: 0.3353 - acc: 0.8616
Epoch 53/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3348 - acc: 0.8614
Epoch 54/100
8000/8000 [==============================] - 2s 261us/step - loss: 0.3348 - acc: 0.8635
Epoch 55/100
8000/8000 [==============================] - 2s 255us/step - loss: 0.3351 - acc: 0.8624
Epoch 56/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.3344 - acc: 0.8621
Epoch 57/100
8000/8000 [==============================] - 2s 216us/step - loss: 0.3346 - acc: 0.8624
Epoch 58/100
8000/8000 [==============================] - 2s 223us/step - loss: 0.3337 - acc: 0.8602
Epoch 59/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.3327 - acc: 0.8625
Epoch 60/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.3338 - acc: 0.8624
Epoch 61/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.3339 - acc: 0.8624
Epoch 62/100
8000/8000 [==============================] - 2s 264us/step - loss: 0.3335 - acc: 0.8614
Epoch 63/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3332 - acc: 0.8642
Epoch 64/100
8000/8000 [==============================] - 2s 240us/step - loss: 0.3335 - acc: 0.8627
Epoch 65/100
8000/8000 [==============================] - 2s 217us/step - loss: 0.3328 - acc: 0.8646
Epoch 66/100
8000/8000 [==============================] - 2s 227us/step - loss: 0.3332 - acc: 0.8621
Epoch 67/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.3332 - acc: 0.8635
Epoch 68/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.3332 - acc: 0.8630
Epoch 69/100
8000/8000 [==============================] - 2s 216us/step - loss: 0.3320 - acc: 0.8640
Epoch 70/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3330 - acc: 0.8626
Epoch 71/100
8000/8000 [==============================] - 2s 260us/step - loss: 0.3327 - acc: 0.8644
Epoch 72/100
8000/8000 [==============================] - 2s 252us/step - loss: 0.3328 - acc: 0.8612
```

```
Epoch 73/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3330 - acc: 0.8621
Epoch 74/100
8000/8000 [==============================] - 2s 229us/step - loss: 0.3321 - acc: 0.8619
Epoch 75/100
8000/8000 [==============================] - 2s 217us/step - loss: 0.3326 - acc: 0.8619
Epoch 76/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3328 - acc: 0.8630
Epoch 77/100
8000/8000 [==============================] - 2s 219us/step - loss: 0.3327 - acc: 0.8631
Epoch 78/100
8000/8000 [==============================] - 2s 219us/step - loss: 0.3317 - acc: 0.8634
Epoch 79/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3326 - acc: 0.8651
Epoch 80/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3325 - acc: 0.8631
Epoch 81/100
8000/8000 [==============================] - 2s 247us/step - loss: 0.3327 - acc: 0.8611
Epoch 82/100
8000/8000 [==============================] - 2s 209us/step - loss: 0.3325 - acc: 0.8639
Epoch 83/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.3328 - acc: 0.8631
Epoch 84/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.3318 - acc: 0.8665
Epoch 85/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3328 - acc: 0.8631
Epoch 86/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.3323 - acc: 0.8635
Epoch 87/100
8000/8000 [==============================] - 2s 215us/step - loss: 0.3325 - acc: 0.8637
Epoch 88/100
8000/8000 [==============================] - 2s 269us/step - loss: 0.3321 - acc: 0.8654
Epoch 89/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3328 - acc: 0.8629
Epoch 90/100
8000/8000 [==============================] - 2s 240us/step - loss: 0.3326 - acc: 0.8630
Epoch 91/100
8000/8000 [==============================] - 2s 214us/step - loss: 0.3317 - acc: 0.8647
Epoch 92/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.3325 - acc: 0.8615
Epoch 93/100
8000/8000 [==============================] - 2s 216us/step - loss: 0.3319 - acc: 0.8651
Epoch 94/100
8000/8000 [==============================] - 2s 216us/step - loss: 0.3323 - acc: 0.8646
Epoch 95/100
8000/8000 [==============================] - 2s 209us/step - loss: 0.3324 - acc: 0.8630
Epoch 96/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3325 - acc: 0.8647
```

```
Epoch 97/100
8000/8000 [==============================] - 2s 257us/step - loss: 0.3318 - acc: 0.8632
Epoch 98/100
8000/8000 [==============================] - 2s 260us/step - loss: 0.3326 - acc: 0.8620
Epoch 99/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.3320 - acc: 0.8631
Epoch 100/100
8000/8000 [==============================] - 2s 222us/step - loss: 0.3321 - acc: 0.8635
```

Out[107]: <keras.callbacks.History at 0x1a0821e3ef0>

In [108]:
```
#Predicting the target values using the model on testing data
y_pred1 = classifier_1.predict(X_test)
y_pred1 = (y_pred1 > 0.5)
```

In [109]:
```
#Creating the confusion matrix
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, y_pred1)
print(cm1)
```

```
[[1513   82]
 [ 194  211]]
```

In [110]:
```
#Calculating the accuracy of the model
accuracy_ANN1 = (cm1[0,0]+cm1[1,1])/ (cm1[0,0]+cm1[0,1]+cm1[1,0]+cm1[1,1])
accuracy_ANN1
```

Out[110]: 0.862

In [111]:
```
#Calculating the specificity of the model
specificity_ANN1 =(cm1[1,1]/(cm1[1,1]+cm1[0,1]))
specificity_ANN1
```

Out[111]: 0.7201365187713311

In [112]:
```
#Calculating the sensitivity of the model
sensitivity_ANN1 =(cm1[0,0]/(cm1[0,0]+cm1[1,0]))
sensitivity_ANN1
```

Out[112]: 0.8863503222026948

In [113]:
```
#Calculating the loss function of the model
classifier_1.evaluate(X_test,y_pred1, verbose=1)
```

```
2000/2000 [==============================] - 0s 181us/step
```

Out[113]: [0.17338521599769594, 1.0]

1.4.3 Model 3

We now create the ANN with two hidden layer and having 6 nodes (Half of feature and target variables) with activation function of the hidden and the input layers as 'tanh' (hyperbolic tangent) and the output layer 'sigmoid' as we need binary result for our classification model. Then we compile the model with SGD 'adam' as its the best which can be used in this case.

```
In [115]: #Creating a Neural Network model
          classifier_2 = Sequential()
          classifier_2.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh', input_d
          classifier_2.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh'))
          classifier_2.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh'))
          classifier_2.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
          classifier_2.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['ac
          classifier_2.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

```
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Update your `

C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: Update your `
  after removing the cwd from sys.path.
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: UserWarning: Update your `
  """
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: UserWarning: The `nb_epoch
  import sys


Epoch 1/100
8000/8000 [==============================] - 8s 986us/step - loss: 0.4787 - acc: 0.8046
Epoch 2/100
8000/8000 [==============================] - 2s 272us/step - loss: 0.4353 - acc: 0.8164
Epoch 3/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.4341 - acc: 0.8129
Epoch 4/100
8000/8000 [==============================] - 2s 227us/step - loss: 0.4310 - acc: 0.8129
Epoch 5/100
8000/8000 [==============================] - 2s 223us/step - loss: 0.4257 - acc: 0.8171
Epoch 6/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.4179 - acc: 0.8239
Epoch 7/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.4103 - acc: 0.8301
Epoch 8/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3993 - acc: 0.8360
Epoch 9/100
8000/8000 [==============================] - 2s 286us/step - loss: 0.3805 - acc: 0.8436
Epoch 10/100
8000/8000 [==============================] - 2s 285us/step - loss: 0.3644 - acc: 0.8511
Epoch 11/100
```

```
8000/8000 [==============================] - 2s 258us/step - loss: 0.3541 - acc: 0.8560
Epoch 12/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3523 - acc: 0.8549
Epoch 13/100
8000/8000 [==============================] - 2s 247us/step - loss: 0.3511 - acc: 0.8590
Epoch 14/100
8000/8000 [==============================] - 2s 233us/step - loss: 0.3488 - acc: 0.8582
Epoch 15/100
8000/8000 [==============================] - 2s 245us/step - loss: 0.3490 - acc: 0.8596
Epoch 16/100
8000/8000 [==============================] - 2s 250us/step - loss: 0.3478 - acc: 0.8609
Epoch 17/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.3477 - acc: 0.8587
Epoch 18/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.3478 - acc: 0.8606
Epoch 19/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3477 - acc: 0.8565
Epoch 20/100
8000/8000 [==============================] - 2s 233us/step - loss: 0.3464 - acc: 0.8601
Epoch 21/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.3467 - acc: 0.8597
Epoch 22/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.3468 - acc: 0.8610
Epoch 23/100
8000/8000 [==============================] - 2s 238us/step - loss: 0.3459 - acc: 0.8594
Epoch 24/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3455 - acc: 0.8589
Epoch 25/100
8000/8000 [==============================] - 2s 287us/step - loss: 0.3454 - acc: 0.8595
Epoch 26/100
8000/8000 [==============================] - 2s 280us/step - loss: 0.3451 - acc: 0.8595
Epoch 27/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3434 - acc: 0.8612
Epoch 28/100
8000/8000 [==============================] - 2s 245us/step - loss: 0.3442 - acc: 0.8605
Epoch 29/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.3438 - acc: 0.8622
Epoch 30/100
8000/8000 [==============================] - 2s 252us/step - loss: 0.3438 - acc: 0.8587
Epoch 31/100
8000/8000 [==============================] - 2s 252us/step - loss: 0.3450 - acc: 0.8587
Epoch 32/100
8000/8000 [==============================] - 2s 282us/step - loss: 0.3440 - acc: 0.8586
Epoch 33/100
8000/8000 [==============================] - 2s 279us/step - loss: 0.3434 - acc: 0.8622
Epoch 34/100
8000/8000 [==============================] - 2s 267us/step - loss: 0.3424 - acc: 0.8630
Epoch 35/100
```

```
8000/8000 [==============================] - 2s 240us/step - loss: 0.3435 - acc: 0.8596
Epoch 36/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.3420 - acc: 0.8607
Epoch 37/100
8000/8000 [==============================] - 2s 234us/step - loss: 0.3441 - acc: 0.8615
Epoch 38/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3432 - acc: 0.8626
Epoch 39/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.3428 - acc: 0.8621
Epoch 40/100
8000/8000 [==============================] - 2s 277us/step - loss: 0.3423 - acc: 0.8614
Epoch 41/100
8000/8000 [==============================] - 3s 316us/step - loss: 0.3427 - acc: 0.8627
Epoch 42/100
8000/8000 [==============================] - 2s 292us/step - loss: 0.3423 - acc: 0.8635
Epoch 43/100
8000/8000 [==============================] - 2s 219us/step - loss: 0.3415 - acc: 0.8610 1s - lo
Epoch 44/100
8000/8000 [==============================] - 2s 220us/step - loss: 0.3429 - acc: 0.8612
Epoch 45/100
8000/8000 [==============================] - ETA: 0s - loss: 0.3418 - acc: 0.861 - 2s 251us/ste
Epoch 46/100
8000/8000 [==============================] - 2s 272us/step - loss: 0.3417 - acc: 0.8609
Epoch 47/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3417 - acc: 0.8617
Epoch 48/100
8000/8000 [==============================] - 2s 274us/step - loss: 0.3422 - acc: 0.8637
Epoch 49/100
8000/8000 [==============================] - 2s 223us/step - loss: 0.3419 - acc: 0.8610
Epoch 50/100
8000/8000 [==============================] - 2s 202us/step - loss: 0.3421 - acc: 0.8605
Epoch 51/100
8000/8000 [==============================] - 1s 177us/step - loss: 0.3401 - acc: 0.8620
Epoch 52/100
8000/8000 [==============================] - 1s 177us/step - loss: 0.3404 - acc: 0.8625
Epoch 53/100
8000/8000 [==============================] - 2s 197us/step - loss: 0.3415 - acc: 0.8616
Epoch 54/100
8000/8000 [==============================] - 2s 262us/step - loss: 0.3407 - acc: 0.8614
Epoch 55/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.3407 - acc: 0.8615
Epoch 56/100
8000/8000 [==============================] - 2s 230us/step - loss: 0.3405 - acc: 0.8611
Epoch 57/100
8000/8000 [==============================] - 2s 268us/step - loss: 0.3402 - acc: 0.8612
Epoch 58/100
8000/8000 [==============================] - 2s 279us/step - loss: 0.3404 - acc: 0.8617
Epoch 59/100
```

```
8000/8000 [==============================] - 2s 255us/step - loss: 0.3397 - acc: 0.8641
Epoch 60/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3392 - acc: 0.8611
Epoch 61/100
8000/8000 [==============================] - 2s 235us/step - loss: 0.3393 - acc: 0.8634
Epoch 62/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.3395 - acc: 0.8637
Epoch 63/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3382 - acc: 0.8637
Epoch 64/100
8000/8000 [==============================] - 2s 229us/step - loss: 0.3382 - acc: 0.8639
Epoch 65/100
8000/8000 [==============================] - 2s 265us/step - loss: 0.3381 - acc: 0.8616
Epoch 66/100
8000/8000 [==============================] - 2s 266us/step - loss: 0.3382 - acc: 0.8605
Epoch 67/100
8000/8000 [==============================] - 2s 250us/step - loss: 0.3378 - acc: 0.8617
Epoch 68/100
8000/8000 [==============================] - 2s 229us/step - loss: 0.3379 - acc: 0.8611
Epoch 69/100
8000/8000 [==============================] - 2s 235us/step - loss: 0.3374 - acc: 0.8636
Epoch 70/100
8000/8000 [==============================] - 2s 233us/step - loss: 0.3368 - acc: 0.8611
Epoch 71/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3369 - acc: 0.8610
Epoch 72/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.3368 - acc: 0.8611
Epoch 73/100
8000/8000 [==============================] - 3s 342us/step - loss: 0.3365 - acc: 0.8627
Epoch 74/100
8000/8000 [==============================] - 2s 276us/step - loss: 0.3356 - acc: 0.8620
Epoch 75/100
8000/8000 [==============================] - 2s 258us/step - loss: 0.3364 - acc: 0.8607
Epoch 76/100
8000/8000 [==============================] - 2s 229us/step - loss: 0.3361 - acc: 0.8615
Epoch 77/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3354 - acc: 0.8639
Epoch 78/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.3355 - acc: 0.8629
Epoch 79/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.3349 - acc: 0.8635
Epoch 80/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.3351 - acc: 0.8617
Epoch 81/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3348 - acc: 0.8627
Epoch 82/100
8000/8000 [==============================] - 2s 275us/step - loss: 0.3348 - acc: 0.8627
Epoch 83/100
```

```
8000/8000 [==============================] - 2s 277us/step - loss: 0.3355 - acc: 0.8626
Epoch 84/100
8000/8000 [==============================] - 2s 235us/step - loss: 0.3353 - acc: 0.8631
Epoch 85/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.3339 - acc: 0.8641
Epoch 86/100
8000/8000 [==============================] - 2s 252us/step - loss: 0.3343 - acc: 0.8652
Epoch 87/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3348 - acc: 0.8634
Epoch 88/100
8000/8000 [==============================] - 2s 231us/step - loss: 0.3331 - acc: 0.8636
Epoch 89/100
8000/8000 [==============================] - 2s 254us/step - loss: 0.3347 - acc: 0.8627
Epoch 90/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.3341 - acc: 0.8651
Epoch 91/100
8000/8000 [==============================] - 2s 262us/step - loss: 0.3330 - acc: 0.8656
Epoch 92/100
8000/8000 [==============================] - 2s 225us/step - loss: 0.3335 - acc: 0.8656
Epoch 93/100
8000/8000 [==============================] - 2s 227us/step - loss: 0.3340 - acc: 0.8642
Epoch 94/100
8000/8000 [==============================] - 2s 224us/step - loss: 0.3338 - acc: 0.8626
Epoch 95/100
8000/8000 [==============================] - 2s 228us/step - loss: 0.3341 - acc: 0.8615
Epoch 96/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3336 - acc: 0.8626
Epoch 97/100
8000/8000 [==============================] - 2s 239us/step - loss: 0.3326 - acc: 0.8641
Epoch 98/100
8000/8000 [==============================] - 2s 296us/step - loss: 0.3334 - acc: 0.8636
Epoch 99/100
8000/8000 [==============================] - 2s 271us/step - loss: 0.3333 - acc: 0.8626
Epoch 100/100
8000/8000 [==============================] - 2s 232us/step - loss: 0.3321 - acc: 0.8654
```

Out[115]: <keras.callbacks.History at 0x1a083c03d68>

In [116]:
```
#Predicting the model with test set
y_pred2 = classifier_2.predict(X_test)
y_pred2 = (y_pred2 > 0.5)
y_pred2
```

Out[116]:
```
array([[False],
       [False],
       [False],
       ...,
```

```
                [False],
                [False],
                [False]])
```

```
In [117]: #Creating the confusion matrix
          cm2 = confusion_matrix(y_test, y_pred2)
          print(cm2)

          #Accuracy of the model
          accuracy_ANN2 = (cm2[0,0]+cm2[1,1])/ (cm2[0,0]+cm2[0,1]+cm2[1,0]+cm2[1,1])
          accuracy_ANN2
```

```
[[1521   74]
 [ 203  202]]
```

```
Out[117]: 0.8615
```

```
In [118]: #Calculating the specificity of the model
          specificity_ANN2 =(cm2[1,1]/(cm2[1,1]+cm2[0,1]))
          specificity_ANN2
```

```
Out[118]: 0.7318840579710145
```

```
In [119]: #Calculating the sensitivity of the model
          sensitivity_ANN2 =(cm2[0,0]/(cm2[0,0]+cm2[1,0]))
          sensitivity_ANN2
```

```
Out[119]: 0.8822505800464037
```

1.4.4 Model 4

```
In [120]: #Creating the Neural Network model
          classifier_2 = Sequential()
          classifier_2.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh', input_
          classifier_2.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
          classifier_2.add(Dense(output_dim = 6, init = 'uniform', activation = 'tanh'))
          classifier_2.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
          classifier_2.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['a

          #Fitting the model on testing data
          classifier_2.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

```
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: Update your
  after removing the cwd from sys.path.
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: UserWarning: Update your
  """
```

```
C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: UserWarning: Update your

C:\Users\Lenovo\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: UserWarning: The `nb_epoc
  # Remove the CWD from sys.path while we load stuff.


Epoch 1/100
8000/8000 [==============================] - 3s 393us/step - loss: 0.4793 - acc: 0.7941
Epoch 2/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.4314 - acc: 0.7951
Epoch 3/100
8000/8000 [==============================] - 2s 261us/step - loss: 0.4301 - acc: 0.7974
Epoch 4/100
8000/8000 [==============================] - 2s 233us/step - loss: 0.4278 - acc: 0.8022
Epoch 5/100
8000/8000 [==============================] - 2s 270us/step - loss: 0.4226 - acc: 0.8131
Epoch 6/100
8000/8000 [==============================] - 2s 282us/step - loss: 0.4153 - acc: 0.8211
Epoch 7/100
8000/8000 [==============================] - 2s 273us/step - loss: 0.4006 - acc: 0.8326
Epoch 8/100
8000/8000 [==============================] - 2s 234us/step - loss: 0.3702 - acc: 0.8511
Epoch 9/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3553 - acc: 0.8579
Epoch 10/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3501 - acc: 0.8595
Epoch 11/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.3487 - acc: 0.8587
Epoch 12/100
8000/8000 [==============================] - 2s 278us/step - loss: 0.3475 - acc: 0.8596
Epoch 13/100
8000/8000 [==============================] - 3s 317us/step - loss: 0.3459 - acc: 0.8595
Epoch 14/100
8000/8000 [==============================] - 2s 290us/step - loss: 0.3461 - acc: 0.8600
Epoch 15/100
8000/8000 [==============================] - 2s 270us/step - loss: 0.3453 - acc: 0.8584
Epoch 16/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3451 - acc: 0.8594
Epoch 17/100
8000/8000 [==============================] - 2s 230us/step - loss: 0.3444 - acc: 0.8605
Epoch 18/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3437 - acc: 0.8602
Epoch 19/100
8000/8000 [==============================] - 2s 234us/step - loss: 0.3437 - acc: 0.8610
Epoch 20/100
8000/8000 [==============================] - 2s 256us/step - loss: 0.3439 - acc: 0.8620
Epoch 21/100
8000/8000 [==============================] - 2s 293us/step - loss: 0.3420 - acc: 0.8625
```

```
Epoch 22/100
8000/8000 [==============================] - 2s 273us/step - loss: 0.3426 - acc: 0.8615
Epoch 23/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3424 - acc: 0.8605
Epoch 24/100
8000/8000 [==============================] - 2s 238us/step - loss: 0.3410 - acc: 0.8605
Epoch 25/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.3411 - acc: 0.8624
Epoch 26/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3408 - acc: 0.8607
Epoch 27/100
8000/8000 [==============================] - 2s 238us/step - loss: 0.3400 - acc: 0.8610
Epoch 28/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3397 - acc: 0.8610
Epoch 29/100
8000/8000 [==============================] - 2s 282us/step - loss: 0.3400 - acc: 0.8587
Epoch 30/100
8000/8000 [==============================] - 2s 276us/step - loss: 0.3392 - acc: 0.8614
Epoch 31/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3390 - acc: 0.8630
Epoch 32/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3389 - acc: 0.8600
Epoch 33/100
8000/8000 [==============================] - 2s 257us/step - loss: 0.3388 - acc: 0.8635
Epoch 34/100
8000/8000 [==============================] - 2s 246us/step - loss: 0.3381 - acc: 0.8601
Epoch 35/100
8000/8000 [==============================] - 2s 239us/step - loss: 0.3372 - acc: 0.8627
Epoch 36/100
8000/8000 [==============================] - 2s 287us/step - loss: 0.3379 - acc: 0.8621
Epoch 37/100
8000/8000 [==============================] - 2s 282us/step - loss: 0.3378 - acc: 0.8626
Epoch 38/100
8000/8000 [==============================] - 2s 269us/step - loss: 0.3372 - acc: 0.8605
Epoch 39/100
8000/8000 [==============================] - 2s 242us/step - loss: 0.3377 - acc: 0.8626
Epoch 40/100
8000/8000 [==============================] - 2s 253us/step - loss: 0.3362 - acc: 0.8621
Epoch 41/100
8000/8000 [==============================] - 2s 242us/step - loss: 0.3370 - acc: 0.8614
Epoch 42/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.3366 - acc: 0.8634
Epoch 43/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.3356 - acc: 0.8640
Epoch 44/100
8000/8000 [==============================] - 2s 283us/step - loss: 0.3362 - acc: 0.8621
Epoch 45/100
8000/8000 [==============================] - 2s 275us/step - loss: 0.3358 - acc: 0.8616
```

```
Epoch 46/100
8000/8000 [==============================] - 2s 260us/step - loss: 0.3351 - acc: 0.8639
Epoch 47/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.3358 - acc: 0.8631
Epoch 48/100
8000/8000 [==============================] - 2s 254us/step - loss: 0.3359 - acc: 0.8611
Epoch 49/100
8000/8000 [==============================] - 2s 264us/step - loss: 0.3357 - acc: 0.8620
Epoch 50/100
8000/8000 [==============================] - 2s 257us/step - loss: 0.3354 - acc: 0.8635
Epoch 51/100
8000/8000 [==============================] - 2s 257us/step - loss: 0.3356 - acc: 0.8631
Epoch 52/100
8000/8000 [==============================] - 2s 282us/step - loss: 0.3354 - acc: 0.8619
Epoch 53/100
8000/8000 [==============================] - 2s 273us/step - loss: 0.3344 - acc: 0.8624
Epoch 54/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.3348 - acc: 0.8619
Epoch 55/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3347 - acc: 0.8634
Epoch 56/100
8000/8000 [==============================] - 2s 246us/step - loss: 0.3345 - acc: 0.8636
Epoch 57/100
8000/8000 [==============================] - 2s 244us/step - loss: 0.3345 - acc: 0.8616
Epoch 58/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3347 - acc: 0.8615
Epoch 59/100
8000/8000 [==============================] - 2s 253us/step - loss: 0.3334 - acc: 0.8634
Epoch 60/100
8000/8000 [==============================] - 2s 280us/step - loss: 0.3343 - acc: 0.8612
Epoch 61/100
8000/8000 [==============================] - 2s 273us/step - loss: 0.3345 - acc: 0.8622
Epoch 62/100
8000/8000 [==============================] - 2s 240us/step - loss: 0.3341 - acc: 0.8612
Epoch 63/100
8000/8000 [==============================] - 2s 240us/step - loss: 0.3342 - acc: 0.8619
Epoch 64/100
8000/8000 [==============================] - 2s 234us/step - loss: 0.3337 - acc: 0.8642
Epoch 65/100
8000/8000 [==============================] - 2s 237us/step - loss: 0.3334 - acc: 0.8614
Epoch 66/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3332 - acc: 0.8625
Epoch 67/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.3333 - acc: 0.8655
Epoch 68/100
8000/8000 [==============================] - 2s 275us/step - loss: 0.3338 - acc: 0.8630
Epoch 69/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.3338 - acc: 0.8626
```

```
Epoch 70/100
8000/8000 [==============================] - 2s 234us/step - loss: 0.3336 - acc: 0.8637
Epoch 71/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3338 - acc: 0.8625
Epoch 72/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3333 - acc: 0.8637
Epoch 73/100
8000/8000 [==============================] - 2s 239us/step - loss: 0.3333 - acc: 0.8640
Epoch 74/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3335 - acc: 0.8612
Epoch 75/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3336 - acc: 0.8609
Epoch 76/100
8000/8000 [==============================] - 2s 276us/step - loss: 0.3329 - acc: 0.8624
Epoch 77/100
8000/8000 [==============================] - 2s 269us/step - loss: 0.3337 - acc: 0.8619
Epoch 78/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3327 - acc: 0.8626
Epoch 79/100
8000/8000 [==============================] - 2s 251us/step - loss: 0.3331 - acc: 0.8629
Epoch 80/100
8000/8000 [==============================] - 2s 241us/step - loss: 0.3328 - acc: 0.8644
Epoch 81/100
8000/8000 [==============================] - 2s 253us/step - loss: 0.3325 - acc: 0.8615
Epoch 82/100
8000/8000 [==============================] - 2s 274us/step - loss: 0.3333 - acc: 0.8634
Epoch 83/100
8000/8000 [==============================] - 2s 253us/step - loss: 0.3327 - acc: 0.8639
Epoch 84/100
8000/8000 [==============================] - 2s 274us/step - loss: 0.3334 - acc: 0.8640
Epoch 85/100
8000/8000 [==============================] - 2s 281us/step - loss: 0.3327 - acc: 0.8632
Epoch 86/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3328 - acc: 0.8625
Epoch 87/100
8000/8000 [==============================] - 2s 226us/step - loss: 0.3323 - acc: 0.8634
Epoch 88/100
8000/8000 [==============================] - 2s 221us/step - loss: 0.3324 - acc: 0.8647
Epoch 89/100
8000/8000 [==============================] - 2s 243us/step - loss: 0.3317 - acc: 0.8646
Epoch 90/100
8000/8000 [==============================] - 2s 240us/step - loss: 0.3325 - acc: 0.8630
Epoch 91/100
8000/8000 [==============================] - 2s 263us/step - loss: 0.3322 - acc: 0.8656
Epoch 92/100
8000/8000 [==============================] - 2s 297us/step - loss: 0.3325 - acc: 0.8635
Epoch 93/100
8000/8000 [==============================] - 2s 268us/step - loss: 0.3319 - acc: 0.8622
```

```
Epoch 94/100
8000/8000 [==============================] - 2s 249us/step - loss: 0.3319 - acc: 0.8627
Epoch 95/100
8000/8000 [==============================] - 2s 248us/step - loss: 0.3323 - acc: 0.8612
Epoch 96/100
8000/8000 [==============================] - 2s 236us/step - loss: 0.3322 - acc: 0.8630
Epoch 97/100
8000/8000 [==============================] - 2s 240us/step - loss: 0.3324 - acc: 0.8632
Epoch 98/100
8000/8000 [==============================] - 2s 247us/step - loss: 0.3316 - acc: 0.8630
Epoch 99/100
8000/8000 [==============================] - 2s 293us/step - loss: 0.3316 - acc: 0.8636
Epoch 100/100
8000/8000 [==============================] - 2s 278us/step - loss: 0.3314 - acc: 0.8636
```

Out[120]: <keras.callbacks.History at 0x1a084107a20>

In [121]:
```
#Predicting the target value using the model on testing data
y_pred3 = classifier_2.predict(X_test)
y_pred3 = (y_pred3 > 0.5)
```

In [122]:
```
#Creating the confusion matrix
cm3 = confusion_matrix(y_test, y_pred3)
print(cm3)
#Calculating the accuracy of the model
accuracy_ANN3 = (cm3[0,0]+cm3[1,1])/ (cm3[0,0]+cm3[0,1]+cm3[1,0]+cm3[1,1])
accuracy_ANN3
```

```
[[1515   80]
 [ 201  204]]
```

Out[122]: 0.8595

In [123]:
```
#Calculating the Specificity of the model
specificity_ANN3 =(cm3[1,1]/(cm3[1,1]+cm3[0,1]))
specificity_ANN3
```

Out[123]: 0.7183098591549296

In [124]:
```
##Calculating the Sensitivity of the model
sensitivity_ANN3 =(cm3[0,0]/(cm3[0,0]+cm3[1,0]))
sensitivity_ANN3
```

Out[124]: 0.8828671328671329

1.5 Conclusion

The model 2 has the accurate representation of the churn rate of bank customers as we achieved highest accuracy among all the ANN models with varying activation functions and number of hidden layers. It has better accuracy of 86.2% with specificity 72.01% and 88.6% sensitivity and low loss function of 0.173.

The model 2 can provide most accurate classification to the bank for their customers at risk of churning. From that data the bank can analyse the customer's requirement and problems to cater the customers in a better manner and to avoid churn.