UNITEDWORLD INSTITUTE OF TECHNOLOGY(UIT)

Summative Assessment (SA)

Submitted BY
Manav Patel
(Enroll. No.: 20220701061)

Course Code and Title: – 21BSCS24C04 Computer Networks

B.Sc. (Hons.) Computer Science / Data Science / AIML
IV Semester – Dec – May 2024

UIT

Dec/May 2024

Customer Churn Prediction Using Artificial Neural Networks (ANN)

ABSTRACT

The challenge of customer churn which affects many a business in different industries remains an extremely thorny problem for managing enterprises. The only way to involve the effective proactive strategies into the retention context is by the accurate prediction of turnover. In this article, we develop the churn prediction model by using Artificial Neural Networks (ANN). The historical customer data is pre-processed before it is passed through ANNs for training. The model's performance is evaluated upon which recommendations are provided for further improvement.

TABLE OF CONTENTS

1.0 INTRODUCTION

The Customer loyalty or retention is the loss of shoppers or clients within a specified period and attrition or turn-over is the common term associated to this. Being a key problem for companies in many different industries, it has attracted the attention of different organizations. Customer lifetime value is the financial measurement of the long-term worthiness of a customer's relationship with the company compared to the acquisition cost of new customers. The shortcomings of this option, nonetheless, may severely compromise the longevity and profitability of a firm in the long run. This way, firms make it feasible to take action in advance and retain clients ensuring the achievement of sustainable revenue growth.

1.1 PROBLEM STATEMENT

The objective is to develop ANN / Artificial Neural Networks based model which will be able to carry out reasonable client retention predictions. To be stealthily accurate, therefore, with the customers who are most inclined to churning, it involves employing the past customer data that comprise the demographics, transactional history and engagement measures among others. The manifold of complex client data, resolving imbalances between the churn and non-churn groups, ensuring model interpretability, and making scalability to handle large data size are some of the difficulties.

1.2 OBJECTIVE

The main aim is development of a customer churn prediction model by using an artificial neural network (ANN). Found on historical data, the algorithm is aimed at precisely clasifying these consumers who are more likely to drop. Tighter client retention is possible for a business and you can increase average client life time value by using actively targeted retention initiatives after achieving this objective.

2.0 UNDERSTANDING CUSTOMER CHURN PREDICTION

2.1 WHAT IS CUSTOMER CHURN?

According to the logic, customer churn is the name of the process of losing clients or customers through the time. It comes with when clients terminate their dealing with the enterprise, discontinue consumption of its products and services as well as prefer other outlets or alternatives over the original one. The typical way to define customer attrition is to divide the number of clients lost by the total number of customers at the start of the period so the metric will be in a percentage format.

2.2 WHY IS CUSTOMER CHURN PREDICTION IMPORTANT?

One of the key variables of corporate growth and consumer retention strategy is the correct forecast of the clients' churn. Firms lessen the risk and stop customer loss through a proactive practice that apprehends potential churn customers.

2.3 HOW CAN ANN HELP IN PREDICTING CUSTOMER CHURN?

The main machine learning method is artificial neural networks (ANN), which borrow their structure and way of working from human brain. ANNs are the systems that consist of networked nodes that are also called the neurons, and they process and transfer data. They can pick up the thin threads and true patterns that may link attrition to these client populations from the extensive databases available thanks to their analytical capabilities.

The ANNs are flexible because of their competency of evaluating customer-related data with demographics, purchasing patterns, and engagement eco-system, either because they can handle both numerical and categorical data. Churn classifications in past customer data with known results can be fed into the model in order for an ANN to learn the underlying patterns which will help it in predicting churn. After being trained through the ANN model, the system will forecast churn for any new client by looking into their traits and behaviour

3.0 APPROACH

3.1 DATA COLLECTION

Credit Card Customer Churn Prediction which contains 684.86 KB data.

The dataset has 10K records, thus, perfect for our project.

https://www.kaggle.com/datasets/rjmanoj/credit-card-customer-churn-prediction

3.2 PREPROCESSING

Perform the preprocessing and data cleaning operations including identifying outliers or missing values and encoding categorical variables.

• Feature engineering can be a powerful tool to scan the data for relevant information, and build any other features that may be needed.

• To ready the data for model training I would suggest you to apply scaling for the numerical features, and encode categorical ones.

3.3 MODEL ARCHITECTURE

- Among many architecture choices, the determine which one to use needs an attention to be paid towards both features of the dataset and overall goal of the task.
- Design the neural network architecture, where the activation functions, number of layers and their neurons, and regularization methods will be employed.

3.4 TRAINING

• Divide the training and testing datasets of the preprocessed data.

• Based on the training data, it will be necessary to train the ANN model and tune the model parameters by gradient descent and backpropagation.

4.0 PROJECT IMPLEMENTATION

STEP 1: IMPORTING THE LIBRARIES

The initial step is using the libraries and dependency we require in our code base. Sci-Kit Learn or Tensor-Flow are, for instance, in addition Pandas, NumPy, matplotlib, and many others.

These libraries offer the following tools:These libraries offer the following tools:

• NumPy for arranged calculations instead.

• pandas for data cleaning and modifying.

• among all three, matplotlib and seaborn are colorful throw-in for data visualization.

• squik learn for the preprocessing and the model evaluation

• tensorflow and keras will be used for implementing artificial neural networks (ANN) model.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('darkgrid')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

STEP 2: DATA ANALYSIS & VISUALIZATION

In order to acquire insight into the dataset and comprehend the elements that can lead to customer turnover, we must next analyze and display the dataset.
This action entails:

Examining the dataset: To begin our study, load the dataset into a panda DataFrame.

```
# read the dataset
df = pd.read_csv('/kaggle/input/credit-card-customer-churn-prediction/Churn_Mode

# see the first five rows of df
df.head()
```

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

First Five Rows of DataFrame

Examining the dataset's shape: Analyze the dataset's dimensions.

```
# see the shape of df
df.shape
```

Getting the dataset's information: Examine the types of data in every column.

```
# see the information of df
df.info()
```

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

Descriptive statistics of the dataset: To comprehend the distribution of numerical variables, compute fundamental statistics like mean, standard deviation, and quartiles.

```
# see the descriptive statistics of df
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000

Examining the dataset for duplicate values: Find and eliminate any rows that are duplicates.

```
# see the duplicates values in dataset if exist then remove otherwise not.
df.duplicated().sum()
```

Finding null values: Examine the dataset to see if any values are missing, then take the necessary action.

```
# check the null values in the each columns of dataset if exist then remove othe
df.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Removing irrelevant columns: Remove any columns that don't apply to our investigation of churn prediction.

```
# drop the irrelevant columns
df.drop(columns=["RowNumber", "CustomerId", "Surname"], inplace = True)
```

```
# check again the first five rows of df after drop the irrelevant columns
df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Analyze and visualize:

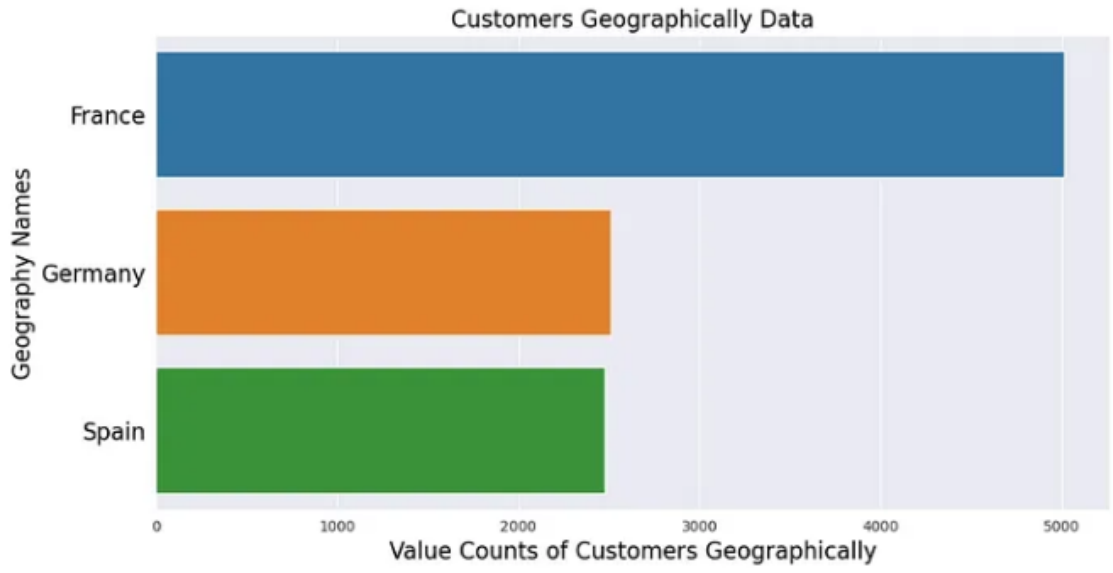
columns like "Geography," "Gender," "Number of Products," "Has Credit Card," "Is Active Member," and the target column "Exited" in order to analyze and visualize those particular columns.

Analysis & Visualization of Geography Feature

```
# check the value counts of "Geography" column.
df_geography = df['Geography'].value_counts()
df_geography
```

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

```
# let's perform univarient eda on "Geography" column.
plt.figure(figsize=(12,6))
res=sns.barplot(x=df_geography, y=df_geography.index)
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 16, color='black')
plt.xlabel('Value Counts of Customers Geographically',fontsize = 16, color='blac
plt.ylabel('Geography Names',fontsize = 16, color='black')
plt.title('Customers Geographically Data',fontsize = 16, color='black')
plt.show()
```

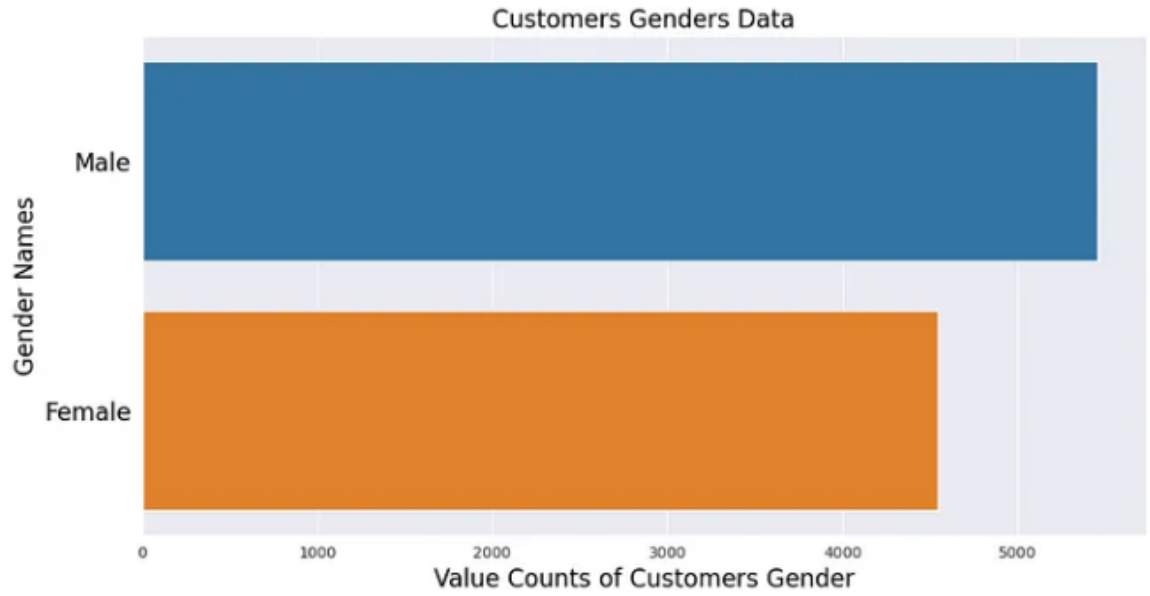


Analysis & Visualization of Gender Feature:

```
# check the value counts of "Gender" column.
df_gender = df['Gender'].value_counts()
df_gender
```

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

```
# let's perform univarient eda on "Gender" column.
plt.figure(figsize=(12,6))
res=sns.barplot(x=df_gender, y=df_gender.index)
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 16, color='black')
plt.xlabel('Value Counts of Customers Gender',fontsize = 16, color='black')
plt.ylabel('Gender Names',fontsize = 16, color='black')
plt.title('Customers Genders Data',fontsize = 16, color='black')
plt.show()
```

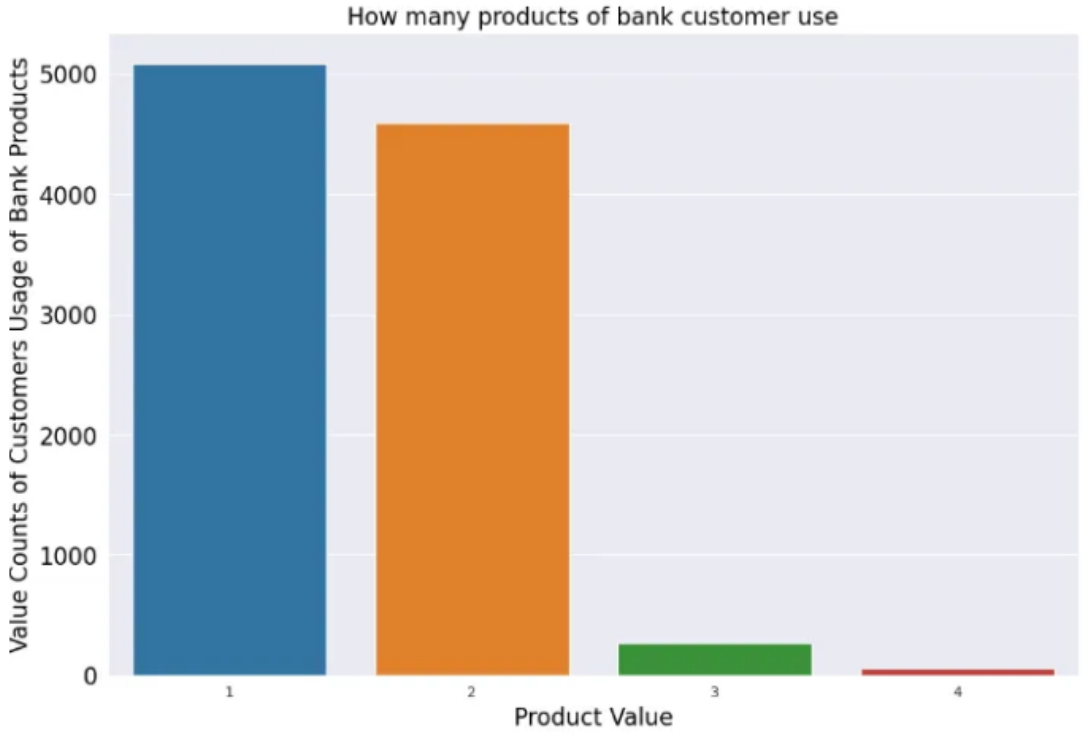


Analysis & Visualization of NumOfProducts Feature:

```
# check the value counts of "number of products" column.
df_nop = df['NumOfProducts'].value_counts()
df_nop
```

```
1    5084
2    4590
3     266
4      60
Name: NumOfProducts, dtype: int64
```

```
# let's perform univarient eda on "number of products" column.
plt.figure(figsize=(12,8))
res=sns.barplot(y=df_nop, x=df_nop.index)
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 16, color='black')
plt.ylabel('Value Counts of Customers Usage of Bank Products',fontsize = 16, col
plt.xlabel('Product Value',fontsize = 16, color='black')
plt.title('How many products of bank customer use',fontsize = 16, color='black')
plt.show()
```



Analysis & Visualization of HasCrCard Feature:

```
# check the value counts of "HasCrCard" column.
df_crc = df['HasCrCard'].value_counts()
df_crc
```

```
1    7055
0    2945
Name: HasCrCard, dtype: int64
```

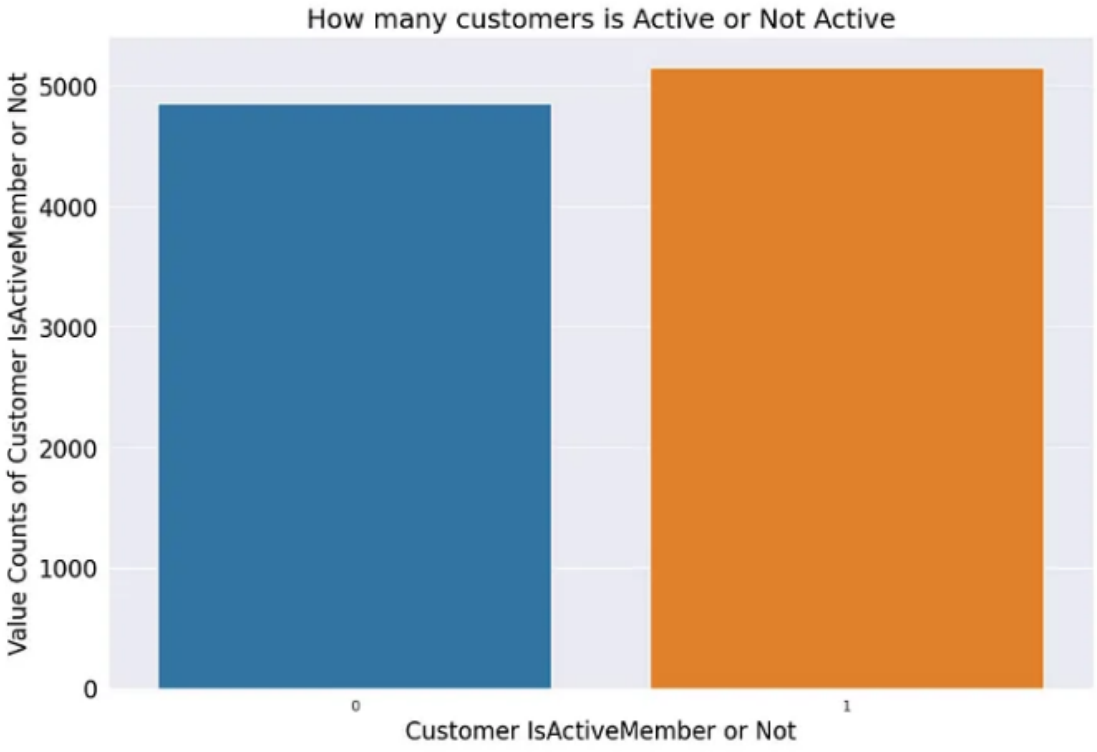
```
# let's perform univarient eda on "HasCrCard" column.
plt.figure(figsize=(12,8))
res=sns.barplot(y=df_crc, x=df_crc.index)
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 16, color='black')
plt.ylabel('Value Counts of Customers Has a Credit Card or Not',fontsize = 16, c
plt.xlabel('Customer have credit card or not',fontsize = 16, color='black')
plt.title('How many customers have credit cards',fontsize = 16, color='black')
plt.show()
```

Analysis & Visualization of IsActiveMember Feature:

```
# check the value counts of "IsActiveMember" column.
df_iam = df['IsActiveMember'].value_counts()
df_iam
```

```
1    5151
0    4849
Name: IsActiveMember, dtype: int64
```

```
# let's perform univarient eda on "IsActiveMember" column.
plt.figure(figsize=(12,8))
res=sns.barplot(y=df_iam, x=df_iam.index)
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 16, color='black')
plt.ylabel('Value Counts of Customer IsActiveMember or Not',fontsize = 16, color
plt.xlabel('Customer IsActiveMember or Not',fontsize = 16, color='black')
plt.title('How many customers is Active or Not Active',fontsize = 18, color='bla
plt.show()
```

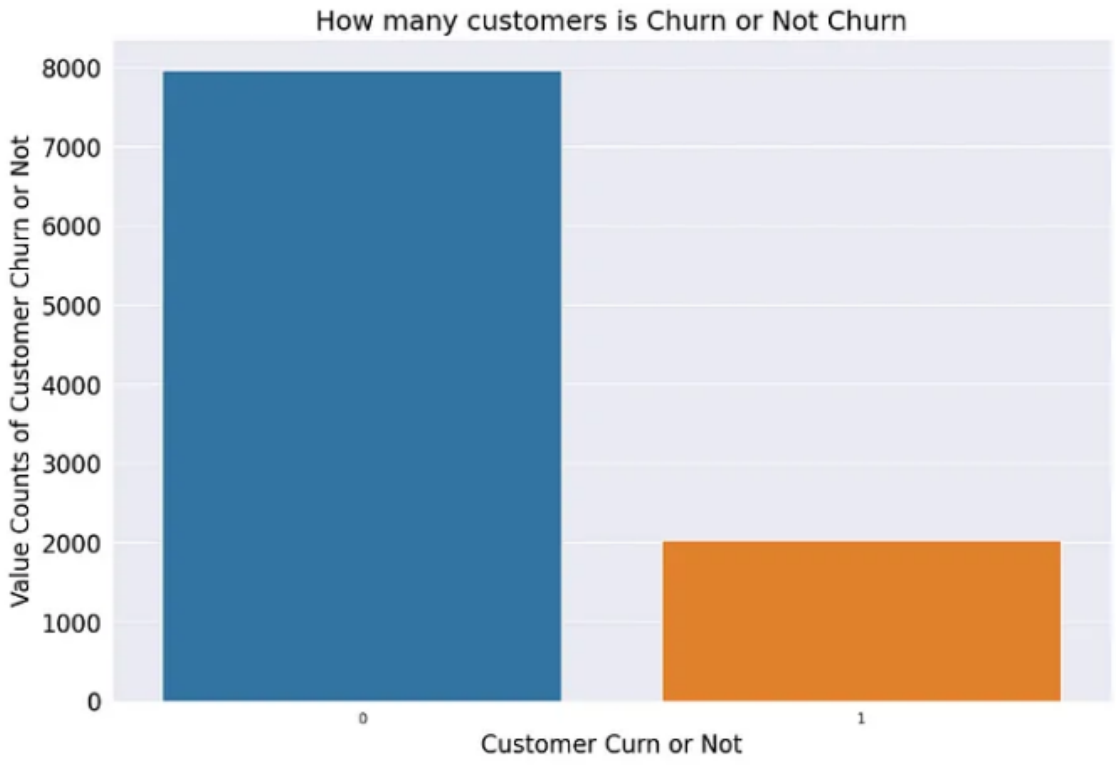


Analysis & Visualization of "Exited" target column:

```
# check the value counts of "Exited" target column.
df_exit = df['Exited'].value_counts()
df_exit
```

```
0    7963
1    2037
Name: Exited, dtype: int64
```

```
# # let's perform univarient eda on "Exited" target column.
plt.figure(figsize=(12,8))
res=sns.barplot(y=df_exit, x=df_exit.index)
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 16, color='black')
plt.ylabel('Value Counts of Customer Churn or Not',fontsize = 16, color='black')
plt.xlabel('Customer Curn or Not',fontsize = 16, color='black')
plt.title('How many customers is Churn or Not Churn',fontsize = 18, color='black
plt.show()
```



STEP 3: APPLYING ONE HOT ENCODING TECHNIQUE

attributes of the dataset which are of category such as “Gender” and “Geography” are should be numerically expressed so that the ANN(Artificial Neural Network) model can process them. The column containing these categories are transformed into the binary columns via the one-hot encoding technique, which stands for various categories.

```
# use pandas dummies funtion for one hot encodeing
df = pd.get_dummies(df, columns=["Gender", "Geography"], drop_first=True)
```

```
# check again the df after one hot encoding method apply
df.head()
```

Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Gender_Male	Geography_Germany	Geography_Spain
0.00	1	1	1	101348.88	1	0	0	0
83807.86	1	0	1	112542.58	0	0	0	1
159660.80	3	1	0	113931.57	1	0	0	0
0.00	2	0	0	93826.63	0	0	0	0
125510.82	1	1	1	79084.10	0	0	0	1

Check again the first five rows of DataFrame after applying the One Hot Encoding Technique

STEP 4: HANDLE THE IMBALANCED TARGET

The datasets with customer churn often exhibits class imbalance, thus having less occasions churner than a non-churner. Minority rows upsampling will maintain the balance in the target column, which is the goal of solving this problem. This ensure that majority is not favored during the training and the model does not show bias due to sample classes.

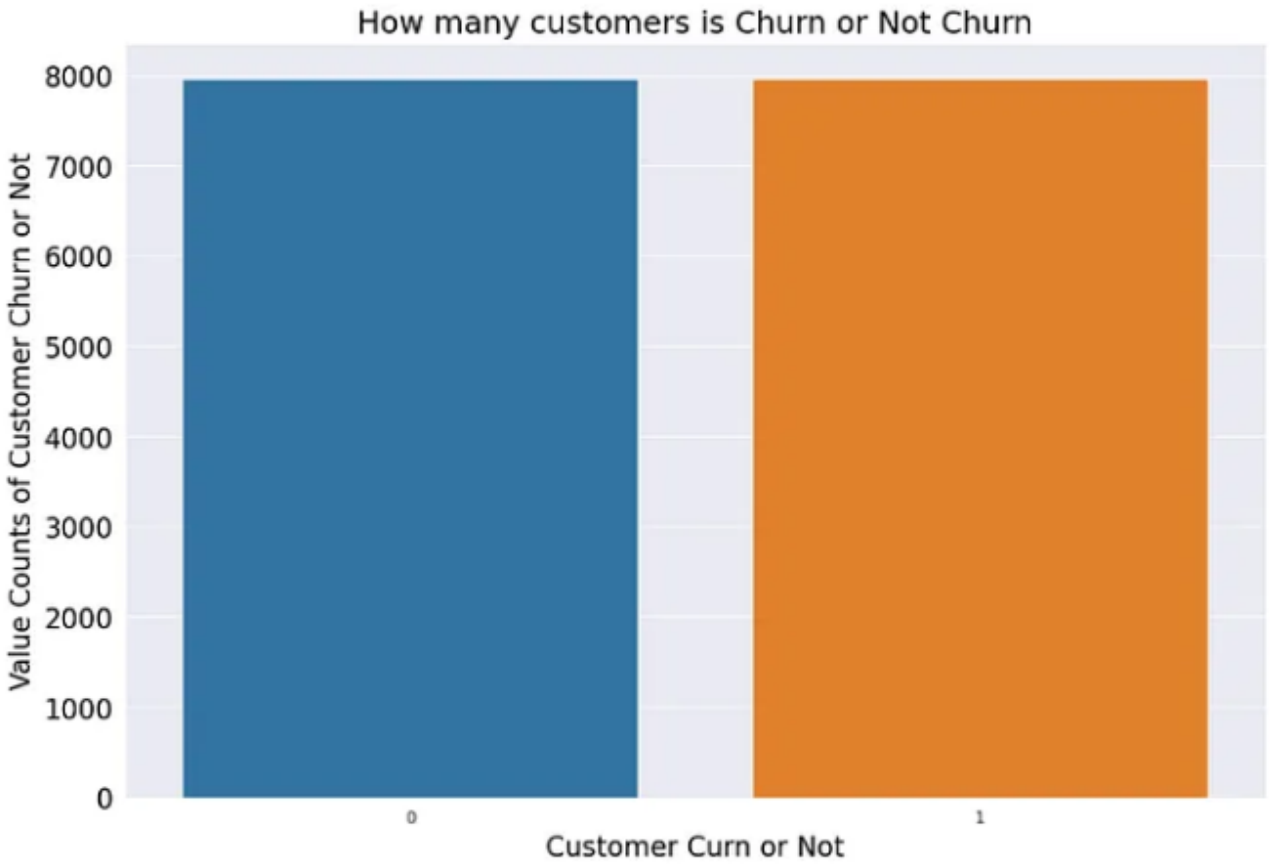
```
# Separate majority and minority classes
majority_class = df[df['Exited'] == 0]
minority_class = df[df['Exited'] == 1]

# Upsample the minority class
upsampled_minority = resample(minority_class,
                              replace=True,  # Sample with replacement
                              n_samples=len(majority_class),  # Match the number
                              random_state=42)  # Set random state for reproduci

# Combine the upsampled minority class with the majority class
balanced_data = pd.concat([majority_class, upsampled_minority])
```

```
# check the target_balanced_data
balanced_data
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Gender_Male
1	608	41	1	83807.86	1	0	1	112542.58	0	0
3	699	39	1	0.00	2	0	0	93826.63	0	0
4	850	43	2	125510.82	1	1	1	79084.10	0	0
6	822	50	7	0.00	2	1	1	10062.80	0	1
8	501	44	4	142051.07	2	0	1	74940.50	0	1
...	...	...	...	...	...	...	...	...	...	...
5701	625	49	4	128504.76	1	1	0	126812.63	1	0
9337	466	47	5	102085.72	1	1	1	183536.24	1	0
4345	720	46	3	97042.60	1	1	1	133516.51	1	1
1085	803	42	5	0.00	1	1	0	196466.83	1	1
3694	608	33	4	0.00	1	1	0	79304.38	1	0

15926 rows × 12 columns

```
# check again the value counts of "Exited" target column after the balancing the
df_exit = balanced_data['Exited'].value_counts()
df_exit
```



STEP 5: SPLITTING THE DATASET INTO TRAINING AND TESTING

Data preprocessing includes feature extraction as well as the splitting of features and the target variable for training our ANN model. The dataset was split into testing and training arms. The test set is applied in order to the model accurary on unknown data, splitting with the data set being used in order to train the model using the historical data. 20-30% of the data (sometimes) are (usually) used for testing, and the rest of it (usually) comprise the training data.

```
# # split the feature in x variable and target varible in y variable
y = balanced_data['Exited']
X = balanced_data.drop('Exited',axis=1)
```

```
# use sklearn for train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
```

```
# check the shape of X_train & X_test, y_train & y_test
print("X_train Shape : ", X_train.shape)
print("X_test Shape : ", X_test.shape)
print("y_train Shape : ", y_train.shape)
print("y_test Shape : ", y_test.shape)
```

```
X_train Shape :  (12740, 11)
X_test Shape :  (3186, 11)
y_train Shape :  (12740,)
y_test Shape :  (3186,)
```

STEP 6: FEATURE SCALING - SKLEARN STANDARD SCALER TECHNIQUE

Feature scaling is a prominent preprocessing step for the ANN models. The same guarantees that their scales are comparable by standardizing the range of input features. Through this step, natural qualities do not overpower each other. Adopting approaches such a Sklearn Standard Scaler Technique for Standardized the Input Features Values, we do the feature scaling.

```
# use sklearn standard scaler technique for standarized the input features value
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# check the X_train_scaled values
X_train_scaled
```

```
array([[ 1.74458075e+00,  2.64948525e-01,  1.02260197e+00, ...,
         9.74118179e-01, -6.65070356e-01, -5.47359263e-01],
       [ 1.24130320e+00,  7.61019204e-02,  8.82175645e-04, ...,
         9.74118179e-01, -6.65070356e-01, -5.47359263e-01],
       [ 8.06835512e-02, -1.83213817e-02,  1.02260197e+00, ...,
         9.74118179e-01,  1.50360032e+00, -5.47359263e-01],
       ...,
       [ 3.16915869e-01, -8.68131101e-01,  8.82175645e-04, ...,
        -1.02656949e+00, -6.65070356e-01, -5.47359263e-01],
       [ 4.60709454e-01, -4.90437892e-01, -3.39691089e-01, ...,
        -1.02656949e+00, -6.65070356e-01, -5.47359263e-01],
       [-2.68529441e-01, -1.12744684e-01,  1.70374850e+00, ...,
         9.74118179e-01, -6.65070356e-01, -5.47359263e-01]])
```

```
# check the X_est_scaled values
X_test_scaled
```

```
array([[ 0.46070945,  0.45379513, -1.02083762, ...,  0.97411818,
        -0.66507036,  1.82695364],
       [-1.34698133, -0.39601459,  1.02260197, ..., -1.02656949,
         -0.66507036, -0.54735926],
       [-1.19291677, -0.39601459, -1.36141088, ...,  0.97411818,
         -0.66507036,  1.82695364],
       ...,
       [-0.11446489, -1.0569777 ,  1.02260197, ...,  0.97411818,
          1.50360032, -0.54735926],
       [-1.0799361 , -0.7737078 ,  1.02260197, ...,  0.97411818,
          1.50360032, -0.54735926],
       [-2.4665171 , -0.39601459, -1.36141088, ...,  0.97411818,
         -0.66507036,  1.82695364]])
```

STEP 7: BUILD THE ANN MODEL

In this step, we draft the architecture of the ANN model which is supported by tensorflow's neural network building API called as Keras. Input, hidden and output layers are among the brain layer ones in layers which are netwoked neurones seen as a whole. Specifying the activation function, the number neurons in each layer and others is another thing. If the issue is complicated, it is quite possible to use a neural network with many layers and many nodes, whereas a little problem can be identified by a small model.

```
model = Sequential()
model.add(Dense(11, activation='relu', input_dim = 11))
model.add(Dense(11, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_23 (Dense)            (None, 11)                132

 dense_24 (Dense)            (None, 11)                132

 dense_25 (Dense)            (None, 1)                 12

=================================================================
Total params: 276
Trainable params: 276
Non-trainable params: 0
_________________________________________________________________
```

STEP 8: TRAINING THE ANN MODEL

After the architecture of the model have been chosen, we proceed to train the ANN model on the dataset for the training. The model modifies its internal representation (weights and indices) throughout the training in order to lower the difference between the predicted and observed churn cases. We set batch size and the number of epochs (passes of the training data set) as training parameters. Some of the valuable metrics that should be tracked might include accuracy loss and validation metrics in order to measure the performance of the model during the training process.

```
model.compile(loss='binary_crossentropy', optimizer='Adam', metrics=['accuracy']
```

```
history = model.fit(X_train_scaled, y_train, epochs = 100, validation_split=0.2)
```

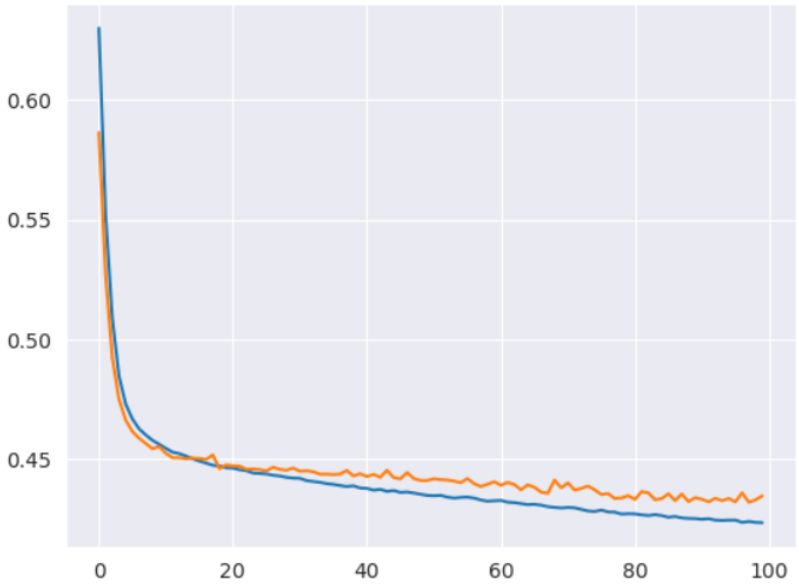
Training Output of ANN Model

```
Epoch 1/100
319/319 [==============================] - 2s 3ms/step - loss: 0.6467 - accuracy: 0.6262 - val_loss: 0.5926 - val_accuracy:
0.6919
Epoch 2/100
319/319 [==============================] - 1s 2ms/step - loss: 0.5775 - accuracy: 0.7002 - val_loss: 0.5646 - val_accuracy:
0.7084
Epoch 3/100
319/319 [==============================] - 1s 2ms/step - loss: 0.5567 - accuracy: 0.7178 - val_loss: 0.5521 - val_accuracy:
0.7206
Epoch 4/100
319/319 [==============================] - 1s 2ms/step - loss: 0.5443 - accuracy: 0.7270 - val_loss: 0.5431 - val_accuracy:
0.7245
Epoch 5/100
319/319 [==============================] - 1s 2ms/step - loss: 0.5326 - accuracy: 0.7333 - val_loss: 0.5306 - val_accuracy:
0.7335

Epoch 96/100
319/319 [==============================] - 1s 2ms/step - loss: 0.4212 - accuracy: 0.8024 - val_loss: 0.4331 - val_accuracy:
0.7995
Epoch 97/100
319/319 [==============================] - 1s 2ms/step - loss: 0.4218 - accuracy: 0.8026 - val_loss: 0.4322 - val_accuracy:
0.7998
Epoch 98/100
319/319 [==============================] - 1s 2ms/step - loss: 0.4213 - accuracy: 0.8005 - val_loss: 0.4321 - val_accuracy:
0.8010
Epoch 99/100
319/319 [==============================] - 1s 2ms/step - loss: 0.4211 - accuracy: 0.8022 - val_loss: 0.4326 - val_accuracy:
0.7959
Epoch 100/100
319/319 [==============================] - 1s 2ms/step - loss: 0.4214 - accuracy: 0.8010 - val_loss: 0.4309 - val_accuracy:
0.8010
```

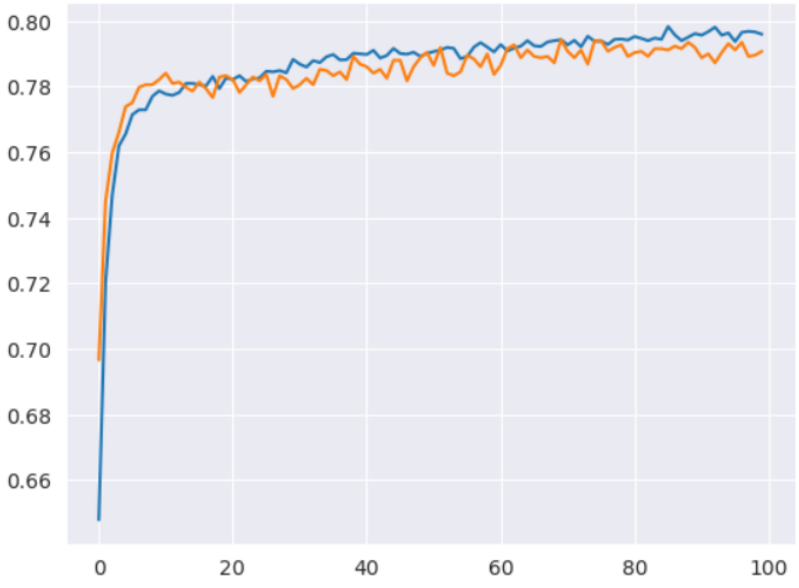
STEP 9: VISUALIZE THE LOSS & ACCURACY OF ANN MODEL

However, the status of the accuracies and loss curves between training and validation data should show how well throughout epochs the model is doing.

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```




```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```



STEP 10: ASSESSING ANN MODEL AND ESTIMATING NEW CUSTOMER CHURN

Employing the testing data, we evaluate the ANN model performance after its training. Assessing the model's success at predicting customer attrition, we calculate a range of metrics including recall, accuracy, precision, and F1 score.

```
y_log = model.predict(X_test_scaled)
```

```
100/100 [==============================] - 0s 1ms/step
```

```
y_pred = np.where(y_log > 0.5, 1, 0)
```

```
accuracy_score(y_test, y_pred)
```

```
0.7944130571249215
```

5.0 CONCLUSION

SUMMARY

The processes required to artificial neural network (ANN) -based model design in order to predict clients' attrition. creating the ANN model with Keras, preprocessing the data, splitting data to the training and testing sets, training and assessing the model, making predictions, and mention some enhancements that might be required.

Customer churn can be predicted using customer churn prediction models which guide a business on how it should react to retain customers. A powerful method for companies to discover and retain those customers who are likely to discontinue the services rendered is the artificial neural network-based customer churn prediction. Institutions can have consumers satisfied, retain both current customer and attract new one through machine learning algorithms and past consumer data.

SOME OPTIMIZATIONS FOR IMPROVING THE MODEL

Hyperparameter tuning: To find the appropriate settings for hyperparameters that will produce models with the best performance, you can experiment with different values for these parameters such as learning rate, batch size, and the number of hidden layers/neurons.

Methods of Regularization: Apply some regularization tricks such as dropout and L1/L2 regularization to attain the model's ability of generalizing new data and prevent overfiting.

Feature Engineering: Search for such new features or form of features that can help to achieve more correct prediction and, in turn, help to have more meaningful data about consumers. The appropriate term for this stage is the feature engineering.

Ensemble Techniques: For ensemble methods like bagging, boosting, and stacking, combine several base versions and therefore take the best of their predictive abilities are for further performance improvement.

6.0 REFERENCES

1. Muhammad Muzzamil. (Jun 8, 2023). Customer Churn Prediction Using Artificial Neural Networks (ANN). Retrieved from: (https://medium.com/@muhammadmuzzamil196/customer-churn-prediction-using-artificial-neural-networks-ann-244450b4d3f8)

2. Ramazan Olmez. (Feb 23, 2024). End-to-End Machine Learning Project: Churn Prediction. Retrieved from: (https://medium.com/@ramazanolmez/end-to-end-machine-learning-project-churn-prediction-258ae4c5b906)

3. Charun Umesh. (Feb 22, 2024). Customer churn prediction with Artificial Neural Network. Retrieved from: https://medium.com/dialog-axiata/customer-churn-prediction-with-artificial-neural-network-d4c46a39cf9)

4. Francesco Franco. (Feb 18, 2024). One-Hot Encoding with Tensorflow and Keras. Retrieved from: (https://medium.com/the-deep-hub/one-hot-encoding-with-tensorflow-and-keras-5b82e059b719)

5. Analytics Vidhya. (Oct 2021). Customer Churn Prediction Using Artificial Neural Network. Retrieved from: https://www.analyticsvidhya.com/blog/2021/10/customer-churn-prediction-using-artificial-neural-network/)