

# CHURN MODELLING

Yash Thakre

Yash Thakre, Under-Graduate, Computer Technology, YCCE college Nagpur, Maharashtra, India

## ABSTRACT

A churn model is a mathematical representation of how churn impacts our business. Churn calculations are built on existing data (the number of customers who left your service during a given time period). A predictive churn model extrapolates on the data to show future potential churn rates. Customer churn is an imperative issue that is frequently connected with the existing cycle of the business. In recent year, churn prediction is becoming a very important issue in Banking and telecom industry. Data set contains details of a bank's customers and the target variable is a binary variable reflecting the fact whether the customer left the bank (closed his account) or he continues to be a customer. This model is used to predict the fact whether the customer left the bank. The dataset is taken from Kaggle Churn modelling. It contains details of a bank's customers and the target variable is a binary variable reflecting the fact whether the customer left the bank (closed his account) or he continues to be a customer. In order to deal with this problem, the industry must recognize these customers before they churn. Therefore, developing a unique classifier that will predict future churns is vital.

## 1) Introduction

The main goal will be exploring the data and predicting the fact whether the customer left the bank using Artificial neural network and also how to perform hyperparameter optimization using KerasTuner.

Dataset : <https://lnkd.in/gY6PKTu>

- 1) RowNumber : Serial number
- 2) CustomerId : Unique Ids for bank customer identification
- 3) Surname : Customer's last name
- 4) CreditScore : Credit score of the customer
- 5) Geography : The country from which the customer belongs
- 6) Gender : Male or Female
- 7) Age : The age of the customer
- 8) Tenure : Number of years for which the customer has been with the bank
- 9) Balance : Bank balance of the customer
- 10) NumOfProducts : Number of bank products the customer is utilising
- 11) HasCrCard : Binary Flag for whether the customer holds a credit card with the bank or not
- 12) IsActiveMember : Binary Flag for whether the customer is an active member with the bank or not
- 13) EstimatedSalary : Estimated salary of the customer in Dollars
- 14) Exited : Binary flag 1 if the customer closed account with bank and 0 if the customer is retained.

RowNum	Customer Surname	CreditScol	Geograph	Gender	Age	Tenure	Balance	NumOfPri	HasCrCard	IsActiveM	Estimated	Exited
1	15634602 Hargrave	619	France	Female	42	2	0	1	1	1	101348.9	1
2	15647311 Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.6	0
3	15619304 Onio	502	France	Female	42	8	159660.8	3	1	0	113931.6	1
4	15701354 Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888 Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012 Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.7	1
7	15592531 Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148 Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.9	1
9	15792365 He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389 H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821 Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173 Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264 Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483 Chin	549	France	Female	25	5	0	2	0	0	190857.8	0
15	15600882 Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0

Fig-1: Representing first 15 Rows with all Columns.

## 2) Keyword

### 1. EDA- Exploratory Data Analysis

In Exploratory Data Analysis (EDA), also known as Data Exploration, is a step in the Data Analysis Process, where a number of techniques are used to better understand the dataset being used. Identifying outliers, missing values, or human error, understanding the relationship(s), or lack of, between variables, ultimately, maximizing your insights of a dataset and minimizing potential error that may occur later in the process.

#### 1.1 Seaborn

Seaborn helps you explore and understand your data. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

#### 1.2 Matplotlib, Plotly

Matplotlib is the most popular data visualization library in Python. It allows us to create figures and plots, and makes it very easy to produce static raster or vector files without the need for any GUI. Plotly, is an open-source library built on top of plotly.js which allows users to create professional quality, web-applications.

### Technologies used



Fig -2: Technologies Used

## 2. Kerastuner

The Keras Tuner is a library that helps you pick the optimal set of hyperparameters for your TensorFlow program. The process of selecting the right set of hyperparameters for your machine learning (ML) application is called hyperparameter tuning or hypertuning. Hyperparameters are the variables that govern the training process and the topology of an ML model. These variables remain constant over the training process and directly impact the performance of your ML program. Hyperparameters are of two types: 1) Model hyperparameters which influence model selection such as the number and width of hidden layers 2) Algorithm hyperparameters which influence the speed and quality of the learning algorithm such as the learning rate for Stochastic Gradient Descent (SGD) and the number of nearest neighbors for K Nearest Neighbors (KNN) classifier

## 3. ANN

Artificial Neural Network(ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems .The network architecture has an input layer, hidden layer (there can be more than 1) and the output layer. It is also called MLP (Multi Layer Perceptron) because of the multiple layers. The hidden layer can be seen as a “distillation layer” that distills some of the important patterns from the inputs and passes it onto the next layer to see. It makes the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information. The activation function serves two notable purposes: 1)It captures non-linear relationship between the inputs 2)It helps convert the input into a more useful output. sigmoid activation function creates an output with values between 0 and 1. There can be other activation functions like Tanh, softmax and RELU. Similarly, the hidden layer leads to the final prediction at the output layer:

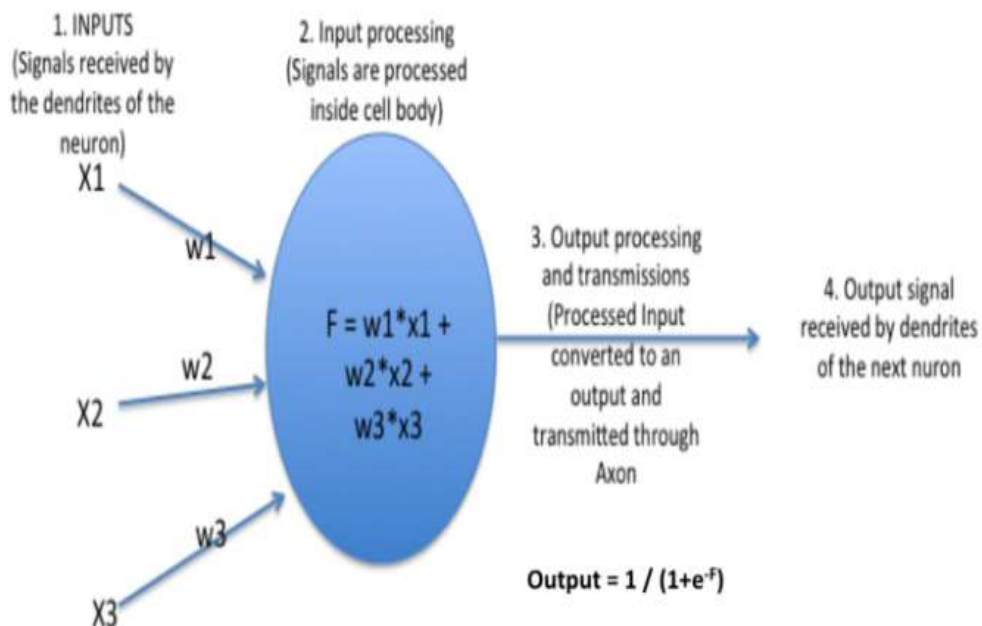


Fig -3: ANN Architecture

### 3. IMPLEMENTATION

#Importing Necessaary Libraries

```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import seaborn as sns
import plotly.graph_objs as px
import plotly.express as ex
from plotly.subplots import make_subplots
from matplotlib import pyplot as plt
%matplotlib inline

# Importing the Keras Libraries and packages
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LeakyReLU, PReLU, ELU
from keras.layers import Dropout
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch

# Feature Scaling and metrics
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
```

Fig -4: Importing Libraries

# Read the csv file

```
data = pd.read_csv('/kaggle/input/churn-modelling/Churn_Modelling.csv')
```

# Exploring the Dataset

- A) Shape of the raw dataset.
- B) Duplicate and null values if any.
- C) Dropping the column which will not make big impact on dependent variables.
- D) Detecting the outliers and removing the outliers data.

A) Shape of the raw data

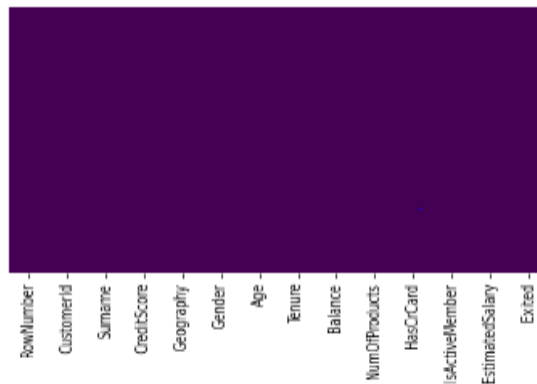
```
print("Shape of Data raw data: {}".format(data.shape))
```

O/P - Shape of Data raw data: (10000, 14)

B) Duplicate null values with heatmap

```
data.duplicated().sum()
```

```
sns.heatmap(data.isnull(),yticklabels = False,cbar = False,cmap = 'viridis')
```



There is no duplicate and missing values in the dataset.

Fig -5: Duplicate null values with heatmap(Output)



```
# Dropping the column which will not make big impact on dependent variables
data.drop(['RowNumber', 'CustomerId', 'Surname'], axis = 1, inplace=True)
```

```
# Stastical analysis of the data
data.describe()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	650.526600	36.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239661	0.203700
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.000000	0.000000	11.580000	0.000000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.000000	0.000000	51002.110000	0.000000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.000000	1.000000	100193.915000	0.000000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.000000	1.000000	149388.247500	0.000000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.000000	1.000000	199992.480000	1.000000

Fig -6: Statistical Analysis of data.

```
# Detecting the outliers
def detect_outlier(data):
    outlier = []
    threshold = 3
    mean = np.mean(data)
    std = np.std(data)
    for i in data:
        z_score = (i - mean)/std
        if np.abs(z_score)>threshold:
            outlier.append(i)
    return outlier

CreditScore_list = data['CreditScore'].tolist()
Balance_list = data['Balance'].tolist()
EstimatedSalary_list = data['EstimatedSalary'].tolist()
CreditScore_outlier = detect_outlier(CreditScore_list)
CreditScore_outlier
Output-[359, 350, 350, 358, 351, 350, 350, 350]

Balance_outlier = detect_outlier(Balance_list)
Balance_outlier
EstimatedSalary_outlier = detect_outlier(EstimatedSalary_list)
EstimatedSalary_outlier

# Shape of Data before removing the outliers
print("Shape of Data before removing outliers: {}".format(data.shape))
Output- Shape of Data before removing outliers: (10000, 11)

# Removing the outliers
data.drop(data[data['CreditScore'] <= 359].index, inplace = True)

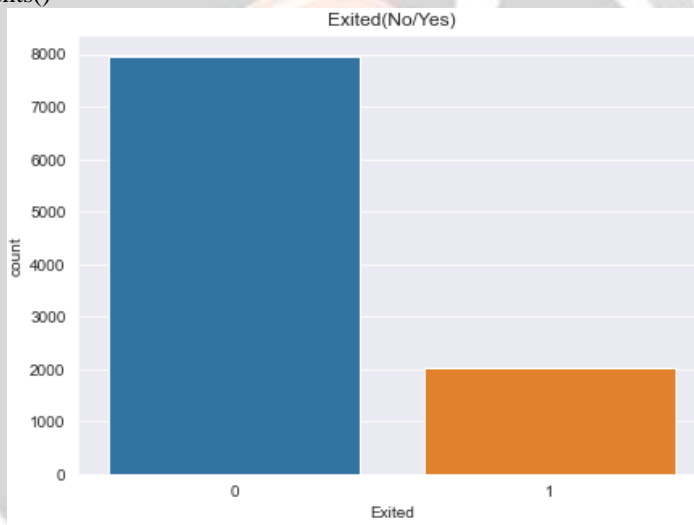
#Shape of Data after removing the outliers
print("Shape of Data after removing outliers: {}".format(data.shape))
Output- Shape of Data after removing outliers: (9992, 11)
```

**#Exploratory Data Analysis**

In this step we are going to plot graphs to get a deeper insights of the data.

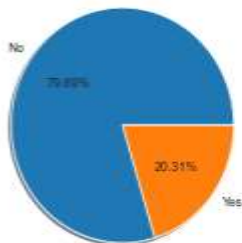
- Proportion of churn vs not churns customers
- Customer churn vs not churns against Gender
- Customer churn vs not churns against country
- Credit card usage according to country
- Credit card usage according to gender
- Country with highest credit score
- Country with highest Estimated salary
- Distribution of Credit score of customer
- Distribution of Customer Age

```
plt.figure(figsize=(7,5))
sns.set_style('darkgrid')
sns.countplot(x='Exited', data=data)
plt.title('Exited(No/Yes)')
data['Exited'].value_counts()
```



**Fig -7:** Bar-graph [ Exited(No/Yes)]

```
plt.pie(data['Exited'].value_counts(), labels=['No', 'Yes'], shadow=True, autopct='%1.2f%%');
```



As we can see from the above graphs, 20.31% of data samples represent the churn customers and 79.69% customer represent the not churn customers.

**Fig -8:** Pie Chart [ Exited(No/Yes)]

```
plt.figure(figsize =(7,5))
sns.set_style('darkgrid')
sns.countplot(x = 'Exited', hue = 'Gender', data = data)
plt.title('Exited against Gender');
pd.DataFrame(data.groupby(['Gender', 'Exited'])['Exited'].count())
```

		Exited
Gender	Exited	
Female	0	3404
	1	1134
Male	0	4559
	1	895

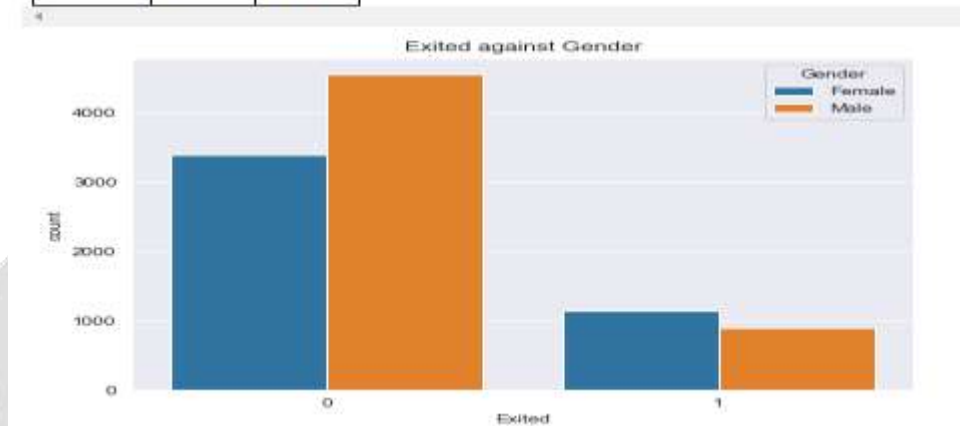


Fig -9: Bar-graph(Exited Against Gender)

```
plt.figure(figsize =(7,5))
sns.set_style('darkgrid')
sns.countplot(x = 'Exited', hue = 'Geography', data = data)
plt.title('Exited against Geography');
pd.DataFrame(data.groupby(['Geography', 'Exited'])['Exited'].count())
```

Out[21]:

		Exited
Geography	Exited	
France	0	4204
	1	806
Germany	0	1695
	1	812
Spain	0	2064
	1	411

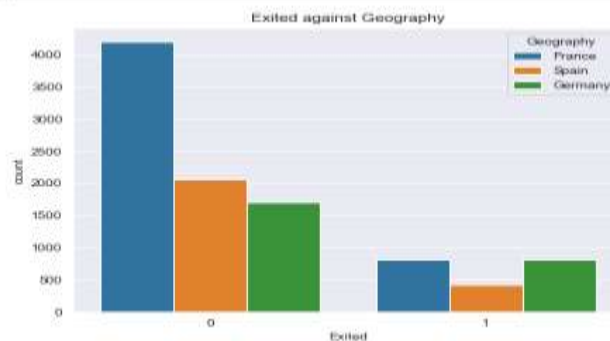


Fig -10: Bar-graph(Exited against Geography)

```
plt.figure(figsize =(7,5))
sns.set_style('darkgrid')
sns.countplot(x = 'HasCrCard', hue = 'Geography', data = data)
plt.title('Credit card used against Geography');
pd.DataFrame(data.groupby(['Geography', 'HasCrCard'])['HasCrCard'].count())
```

		HasCrCard
Geography	HasCrCard	
	0	1
France	0	1470
	1	3540
Germany	0	717
	1	1790
Spain	0	756
	1	1719

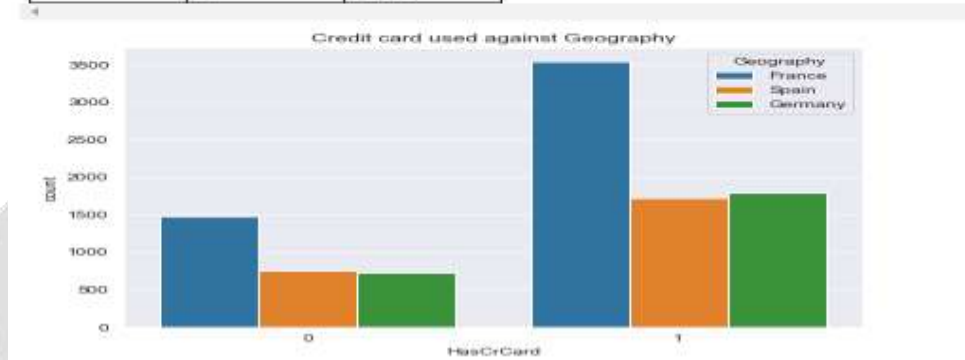


Fig -11: Bar-Graph(Credit Card against against Geography)

```
plt.figure(figsize =(7,5))
sns.set_style('darkgrid')
sns.countplot(x = 'HasCrCard', hue = 'Gender', data = data)
plt.title('Credit card used against Gender');
pd.DataFrame(data.groupby(['Gender', 'HasCrCard'])['HasCrCard'].count())
```

		HasCrCard
Gender	HasCrCard	
	0	1
Female	0	1350
	1	3188
Male	0	1593
	1	3861

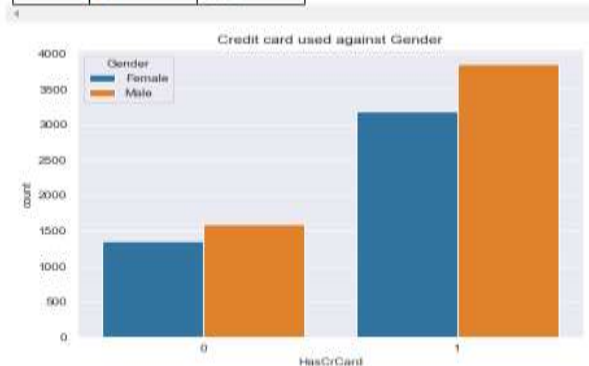


Fig -12:Bar-Graph(Credit Card Used against Gender)



Proportionally more female customers are exited from the bank as compared to male customers.  
 Proportionally more German customers are exited from the bank as compared to other two countries.  
 More number of customers from France have credit card as compared to other two countries.  
 Proportionality of customer having credit card against gender is almost equal.

```
fig = ex.box(data, x="Exited", y="CreditScore", color = 'Geography')
fig.update_layout(title_text="Different Country with mean Credit scores(Exited(No/Yes))")
fig.show();
fig = ex.box(data, x="Exited", y="EstimatedSalary", color = 'Geography')
fig.update_layout(title_text="Different Country with mean salary(Exited(No/Yes))")
fig.show();
```

Credit score and estimated salary does not effect much on exit rates.

```
fig = make_subplots(rows=1, cols=1)
hist=px.Histogram(x=data['CreditScore'],name='Credit Score Histogram')
fig.add_trace(hist,row=1,col=1)
fig.update_layout(height=500, width=700, title_text="Distribution of the Credit score")
fig.show()
fig = make_subplots(rows=1, cols=1)
hist=px.Histogram(x=data['Age'],name='Age Histogram')
fig.add_trace(hist,row=1,col=1)
fig.update_layout(height=500, width=700, title_text="Distribution of the Customer Ages")
fig.show()
```

We can see that distribution of customer ages and credit score in our dataset follows a fairly normal distribution; we can use these features with the normality assumption.

```
corrmat = data.corr() # Correlation between the features using heatmap
plt.figure(figsize=(10,8))
sns.heatmap(corrmat, annot=True, cmap="RdYlGn")<AxesSubplot:> #plot heat map
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x28f24866a30>
```

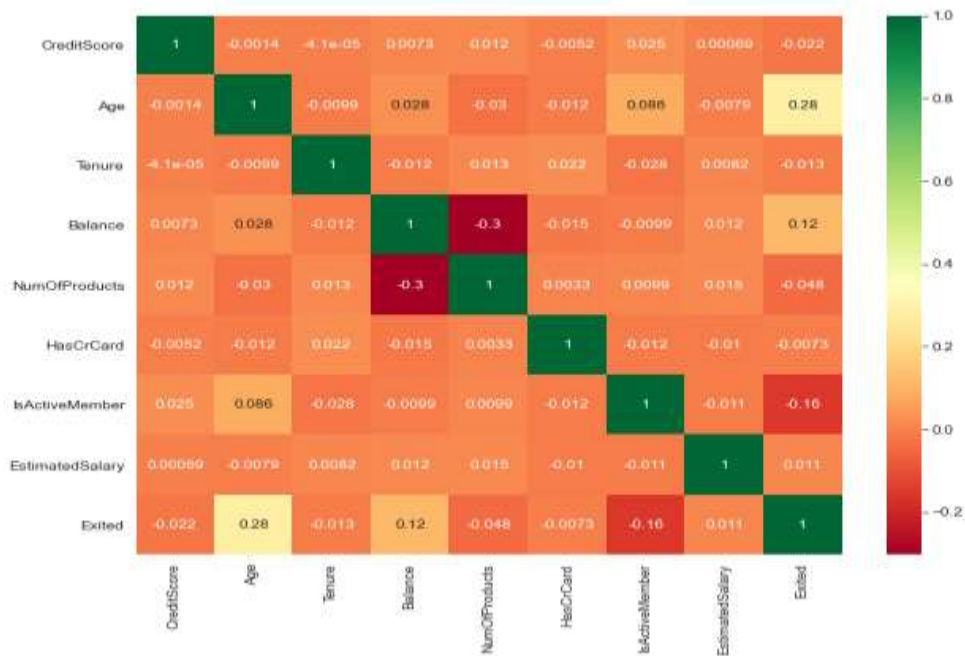


Fig -13: Heat-map features Co-relation

**#Data Preprocessing**

Splitting the dataset into dependent and independent variables.

Creating dummy variables of categorical column using OneHotEncoding.

Determining train and test set.

Feature scaling using standard scaler

```
# Split the Dataset
X= data.drop(['Exited'], axis = 1)
y = data['Exited']

# Creating dummy variables
Dummies = pd.get_dummies(X[['Geography', 'Gender']],drop_first=True)
X = X.drop(['Geography', 'Gender'], axis = 1)
X = pd.concat([X, Dummies], axis = 1)

# Splitting the dataset into the Training set and Test set
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_state = 42)

# Feature Scaling using standard scaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

**#Data Modelling**

Artificial Neural Network (ANN)

# Initialising the ANN

```
classifier = Sequential()
```

# Adding the input layer and the first hidden layer

```
classifier.add(Dense(units = 16, kernel_initializer = 'he_uniform',activation='relu',input_dim = 11))
```

# Adding the second hidden layer

```
classifier.add(Dense(units = 8, kernel_initializer = 'he_uniform',activation='relu'))
```

# Adding the output layer

```
classifier.add(Dense(units = 1, kernel_initializer = 'glorot_uniform', activation = 'sigmoid'))
```

# Compiling the ANN

```
classifier.compile(optimizer = 'Adamax', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

# Fitting the ANN to the Training set

```
model_history=classifier.fit(X_train, y_train,validation_split=0.33, batch_size = 128, epochs = 100)
```

# summarize history for accuracy

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

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

```
plt.title('model accuracy')
```

```
plt.ylabel('accuracy')
```

```
plt.xlabel('epoch')
```

```
plt.legend(['train', 'test'], loc='upper left')
```

```
plt.show()
```

```
# summarize history for loss
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
```

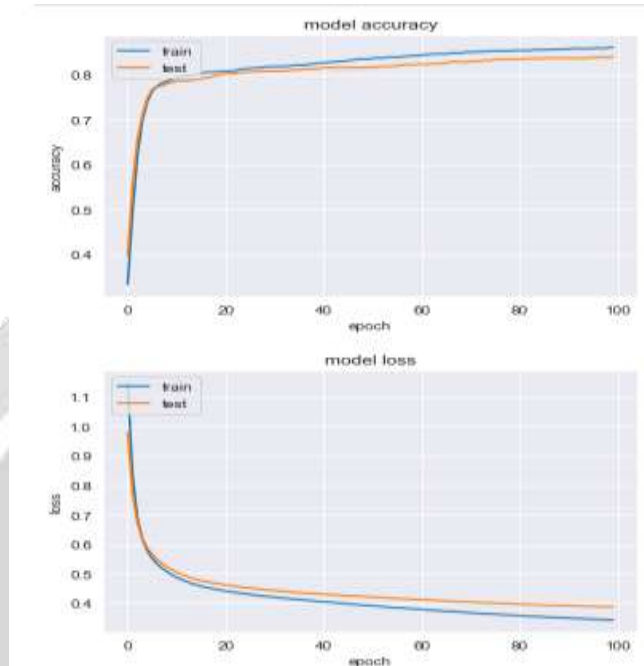


Fig -14: Model Accuracy, Model Loss

```
# Making the predictions and evaluating the model
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
# Making the classification report
print(classification_report(y_test, y_pred))
# Making the Confusion Matrix
print(confusion_matrix(y_test, y_pred))
# Calculate the Accuracy
print(accuracy_score(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.87	0.96	0.91	1603
1	0.71	0.42	0.53	396
accuracy			0.85	1999
macro avg	0.79	0.69	0.72	1999
weighted avg	0.84	0.85	0.84	1999
[[1534 69]				
[ 228 168]]				
0.8514257128564282				

Fig -15: Hyperparameter

### 4. CONCLUSIONS

After performing hyperparameter tuning using keras tuner accuracy, somewhat increased to 0.8634 from 0.8514. Overall, the Keras Tuner library is a nice and easy to learn option to perform hyperparameter tuning for Keras and Tensorflow 2.0 model .Customer churn analysis allows to minimize acquisition costs and increase marketing efficiency and preparing a solid base for future marketing analysis and campaigns. Customer churn analysis opens new opportunities for cross-selling and upselling and serves as one of the starting points for customer-driven product development, keeping customers engaged and loyal over time.

```

model = tuner.get_best_models(num_models=1)[0]

model.fit(X_train, y_train, epochs=200, initial_epoch=6, validation_data=(X_test,y_test), verbose = 0)
<tensorflow.python.keras.callbacks.History at 0x28f2c71e760>


model.summary()
Model: "sequential"
Layer (type)                Output Shape                Param #
-----
dense (Dense)                (None, 96)                  1152
dropout (Dropout)           (None, 96)                  0
dense_1 (Dense)              (None, 40)                  3880
dropout_1 (Dropout)         (None, 40)                  0
dense_2 (Dense)              (None, 1)                   41
-----
Total params: 5,073
Trainable params: 5,073
Non-trainable params: 0

# Evaluate the best model.
loss, accuracy = model.evaluate(X_test, y_test)
63/63 [=====] - 0s 1ms/step - loss: 0.5275 - accuracy: 0.8634
    
```

Fig -16: Hyperparameter Optimization using keras tuner

### 5. REFERENCES

[1] <https://www.coursera.org/learn/machine-learning-with-python/ungradedLti/pUIUL/lab-knn>  
 [2] <https://seaborn.pydata.org/introduction.html>  
 [3] [https://github.com/ ANN/Churn\\_modelling.ipynb](https://github.com/ANN/Churn_modelling.ipynb)  
 [4] [indatalabs.com/blog/customer-churn-analysis-prediction](http://indatalabs.com/blog/customer-churn-analysis-prediction)

	<p><b>YASH THAKRE</b>                  CSE undegrade, YCCE college , Nagpur.                  Likes to bring insights from the data to solve the real world problems.                  Linkdin profile-  <a href="https://www.linkedin.com/in/yash-thakre-b9004216b/">https://www.linkedin.com/in/yash-thakre-b9004216b/</a>                  Email address- yash.thakre63@gmail.com</p>
---	--