Customer Churning Prediction Using ANN

MLND Capstone Project

Mihir Rajput | 14th May, 2020

I. Definition

1.1 Project Overview

The churn rate is the percentage of subscribers to a service who discontinue their subscriptions to the service within a given time period. For a company to expand its clientele, its growth rate, as measured by the number of new customers, must exceed its churn rate.

The goal of Churn prediction is to detect customers is intended to leave a service provider or not. Retaining existing customer costs an organization from 5 to 10 times than gaining a new customer.

Predictive models can provide correct identification of possible churners in the near future in order to provide a retention solution.

So that organizations can predict the insights, loss, gains and plan their strategies. Also in addition this will help a large organization who delivers services to the customers to make important decisions based on predicted insights.

There was a research paper available from the past which was published to tackle these specific types of problems. I have attached its link in a reference section.

1.2 Problem Statement

When I was interning with a Multinational Telecom Company, there the Data Science Team was working on this problem. But their dataset was a huge one. Here I have provided one simple dataset.

So, here the Company wants a model which can predict that how likely its current customers will leave the company in near future and hence calculate its churn rate.
This is a Classification Problem in which you'll classify a customer based on his/her Credit Score, Region, Gender, Age, Tenure, Balance, Salary etc. whether he/she will EXIT(1) or NOT(0).

Using these model predictions company will be aware of upcoming churnings by the customers and they will be able to prevent or reduce its rate as well.

Predictions would be also useful to plan resources consumption and new expansion strategies.

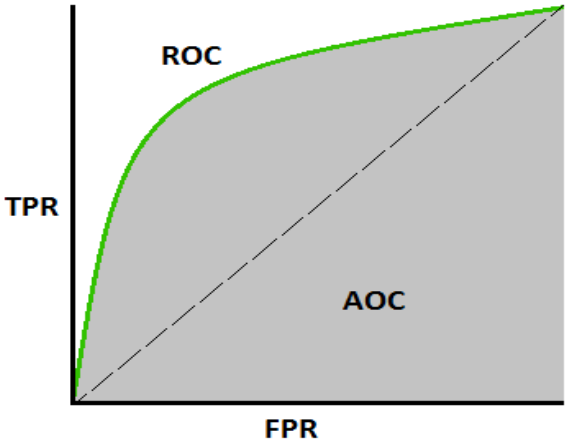
To solve the problem I will be using deep learning. To be specific I will be

using ANN (artificial neural networks) to predict whether the customer will leave the bank or not.

1.3 Metrics

For the evaluation purposes we use ROC-AUC curve, confusion matrix, precision, recall, and F1 scores along with the accuracy.

- **ROC-AUC Curve -** AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.



Defining terms used in AUC and ROC Curve.

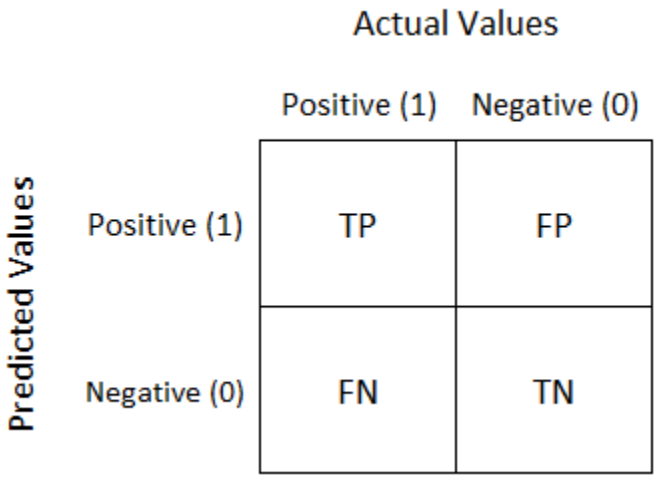
$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{FPR} = 1 - \text{Specificity}$$

$$= \frac{\text{FP}}{\text{TN} + \text{FP}}$$

- **Confusion Matrix** – It is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



- **Precision, Recall and F1 Score –**

  Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

  Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

  F1 Score is the weighted average of Precision and Recall.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

- **Accuracy Score -** Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

Accuracy = TP+TN/TP+FP+FN+TN

II. Analysis

2.1 Data Exploration

The datasets are provided by Kaggle on its official competition website. They are free to download.

This dataset contains customer's data like salary, gender, age, and credit score along with the customer's current status, existed or not. That's why this dataset is very nice match to solve churning problem.

The dataset consist of 10,000 rows, I will be using 80% of them to train the model and 20% to evaluate the model.

This dataset is imbalanced, because class 1(Exited) has 7500 rows and the class 0(Not Existed) has 2500 rows.

Although there are some null entries in the dataset by removing them I might be able to balance the dataset properly.

Input Data fields

- CustomerId – Id of the customer
- Surname – Surname of the customer
- CreditScore – Customer's credit score
- Geography – Geo-location of the customer
- Gender – Gender of the customer
- Age – Age of the customer
- Tenure – Tenure of the customer
- Balance – Customer's current balance
- NumOfProducts – Number of products purchased by the customer
- HasCrCard – 0 if customer has credit card 1 otherwise.
- IsActiveMember – 0 if customer is not active 1 otherwise.
- EstimatedSalary – Estimated salary of the customer.
- Exited – 1 if existed 0 otherwise.

You can find sample dataset below.

RowNumb	Customer	Surname	CreditSco	Geograph	Gender	Age	Tenure	Balance	NumOfPr	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

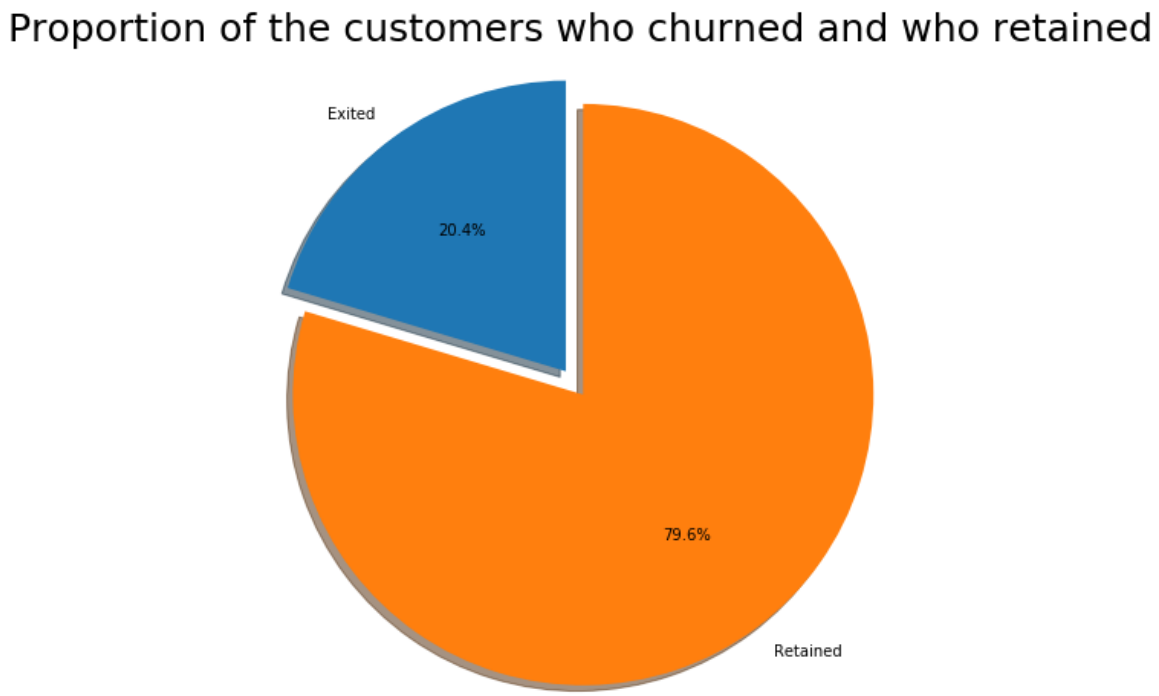
There are no missing values in the dataset as shown below.

Column Name	Null Count
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

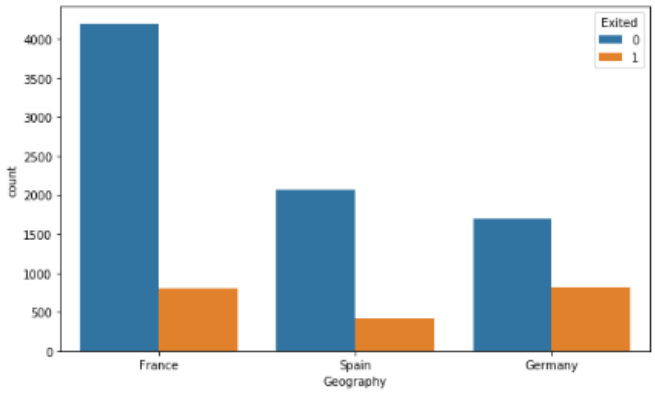
And here is the data type of different columns.

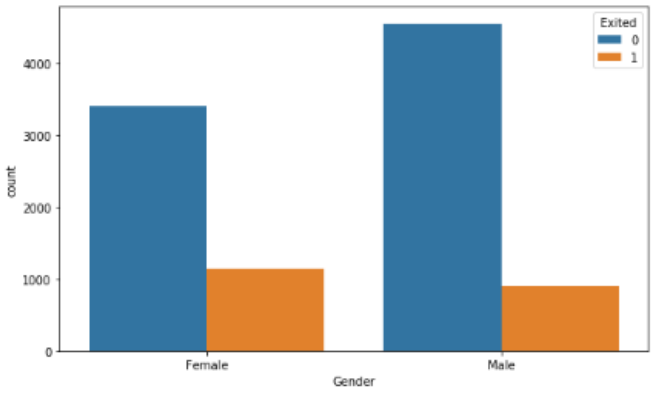
Column Name	Datatype
CreditScore	int64
Geography	object
Gender	object
Age	int64
Tenure	int64
Balance	float64
NumOfProducts	int64
HasCrCard	int64
IsActiveMember	int64
EstimatedSalary	float64
Exited	int64

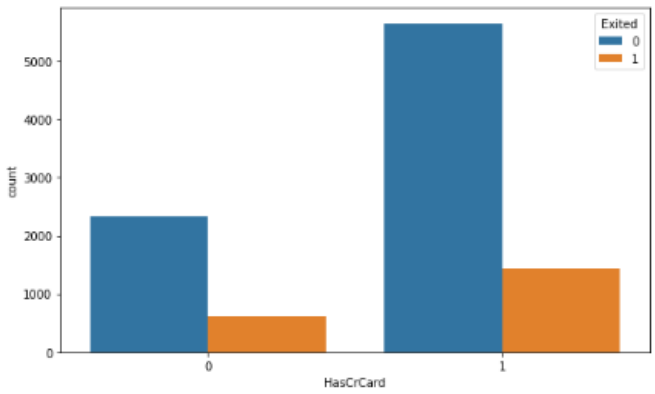
2.2 Exploratory Visualization

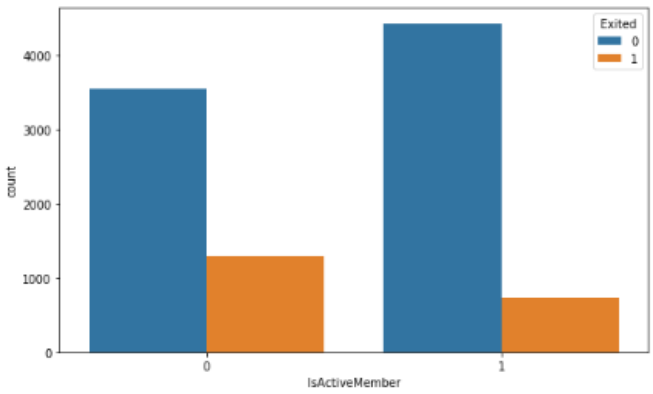


So the graphs show that 20.4% of the customers have exited.

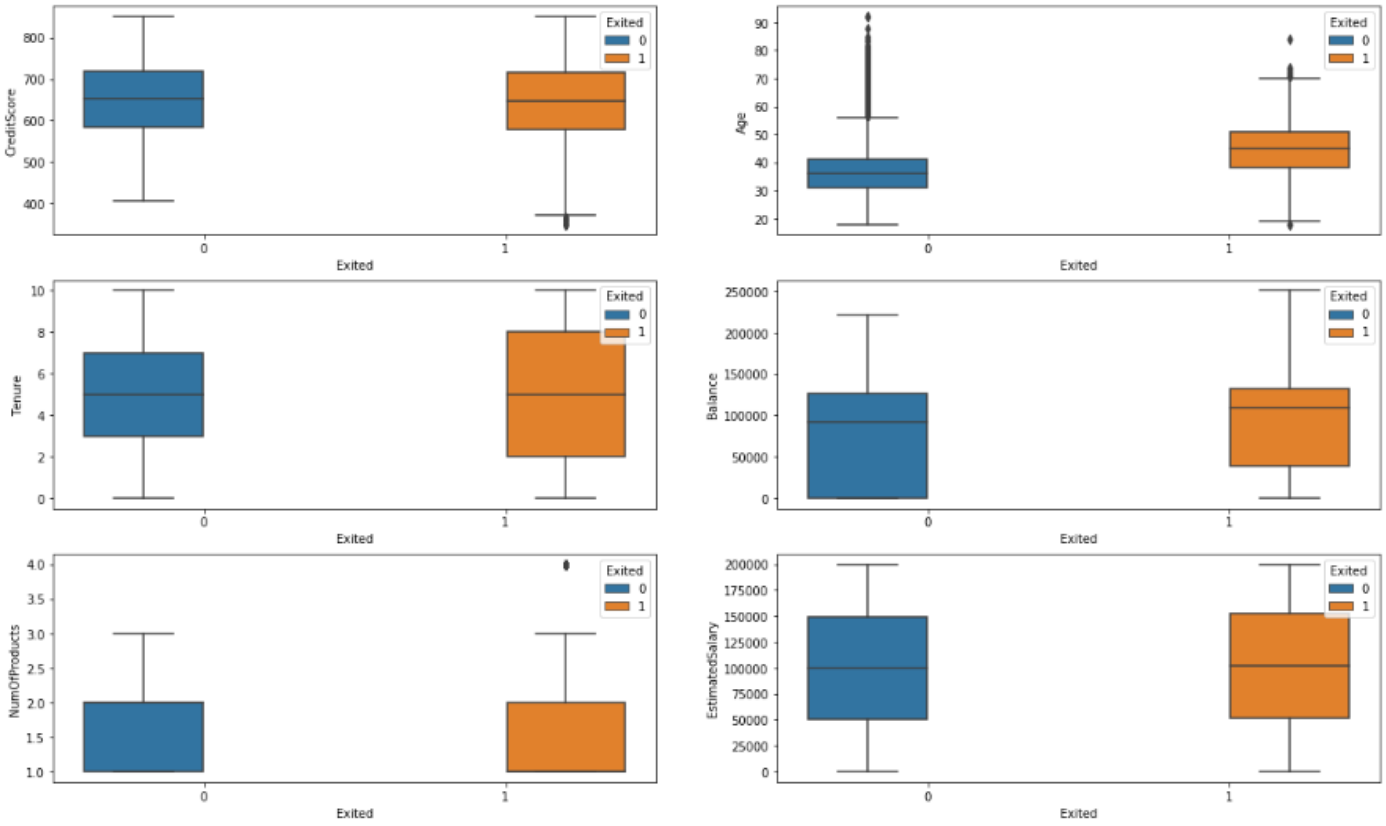








- Majority of the data is from France region.
- The proportion of female customers churning is also greater than that of male customers
- Customer with the credit card is most likely to churn.
- Lastly, the inactive customers have a more churning rate as compared to the active customers.



- There is no big difference in the credit score distributions between exited and retained customers.
- The older customers are churning more than the younger customers.
- There is no big impact of tenure on churning rate.
- Customers with the high bank balance are churning more than low balance.
- There is no big impact of number of purchased product on churning rate.
- There is no big impact of estimated salary on churning rate.

2.3 Algorithms and Techniques

For training I am leveraging the power of the deep learning and neural networks.

First I have performed feature engineering to extract which columns are useful and contributing more in the churning distribution.

Then I have normalized all the data using min-max scaling.

After that I have applied one-hot-encoding on categorical columns to standardize the data.

And finally, for the model training I am using Artificial Neural Networks with 5 hidden layers with different number of neurons. Additional information is attached below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_34 (Dense)             (None, 512)               8704
_________________________________________________________________
dense_35 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_36 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_37 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_38 (Dense)             (None, 16)                1040
_________________________________________________________________
dense_39 (Dense)             (None, 1)                 17
=================================================================
Total params: 182,241
Trainable params: 182,241
Non-trainable params: 0
_________________________________________________________________
```

- Batch-size: 64
- Epochs: 20
- Optimization function: Adam
- Learning rate: 0.001
- Activation function: ReLU for hidden layers and sigmoid for output layer
- Weight initialization: uniform distribution
- Loss function: binary crossentropy

2.4 Benchmark

- Since this is a binary classification problem, the benchmark model will be SVC based model. I will try to outperform this model's accuracy which is near 80%.
- There are some pre-built solutions for this problem, based on machine learning algorithms approach like Decision Tree, Logistic Regression and RandomForest. I will try to compare my results with these algorithms as well.

III. Methodology

3.1 Data Preprocessing

For the data preprocessing I have applied several steps which are mentioned in detail below.

- Data Rearrange

  Here I have rearranged the data. And changed their indexing orders this makes data look in simple manner.

  I have separated the categorical and continuous columns so that it will be easy to manipulation for pre-processing and training.

- One hot encoding

  Then I have applied one hot encoding with -1 to 0 boundaries on **HasCrCard** and **IsActiveMember** columns so that I can use negative impact of these columns on churning distribution.

  Then I have applied one hot encoding on categorical columns like **Geography** and **Gender**.

  One hot encoding helps training algorithm understand the data well and easy to process.

- Scaling

  I also have applied min-max scaling to scale all the continuous variables.

  Scaling is very important and necessary. This prevent model from getting overfit or underfit.

  Also this is helpful for stable training process. It is proven that scaled and standardized data helps optimizers converge faster. Faster convergence leads to accurate model!

- Splitting the data

  After applying all the required pre-processing steps I have created two sets training set and test set.

  However in while training the network I am dividing the training set again into two sets. From training set itself I am creating two sets.

  This makes three splits.

  All the evaluations are derived from validation original testing set. This set is unseen for the model.

1. **Original Training set** (80% entire dataset)
    - **Training set** (80 % of original training set)
    - **Validation set** (20 % of original training set)
2. **Original Testing set** (20% of entire dataset)

3.2 Implementation

- I have used keras with tensorflow backend to implement the artificial neural network.
- The ANN gets 16 features as an input and outputs single value between 0 and 1. If the value is greater than 0.5 then it would be considered as true else false.
- ANN implementation is pretty straight forward and simple. I have used 5 hidden layers with different neuron values.
- For the optimization algorithm I have choose Adam. Because it is fast and helps converge well.
- I have used binary cross entropy loss function, which is based on logloss.
- Hidden Layers has ReLU(Rectified Linear Unit) as an activation function and output layer has sigmoid.
- The training process is also very simple, we will pass data with the batch size of 16 and learning rate is set to 0.001.

3.3 Refinement

- To train the ANN I have tried different experiments and hyper parameter tuning.
- I tried with different number of hidden layers and different number of neurons.
- I experimented with different values for learning rate like 0.1, 000.1 0.005 etc.
- I also tried different activation functions like softmax, tanh and linear.
- I trained the network with different batch sizes like 10, 16, 64, and 128.
- I tried different epoch values for ANN like 10, 30, 50 and 100 even.

IV. Results

4.1 Model Evaluation and Validation

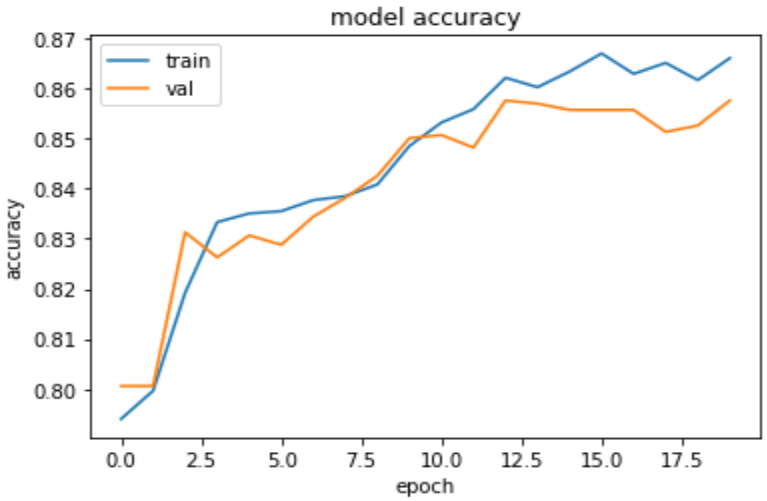
- Model is evaluated with certain hyper parameters after trying number of experiments.
- I tweaked most of the hyper parameters and experimented with them until I got optimal set of the hyper parameters.
- Optimal set of the hyper parameters for this problem is shown below.

Parameter	Value
batch size	64
epochs	20
optimization function	Adam
learning rate	0.001
activation function	ReLU for hidden layers and sigmoid for output layer
weight initialization	uniform distribution
loss function	binary cross-entropy

- I made a split of data one for training and one for testing. The split ratio is 80% for training and 20% for evaluation and validation.
- The 20% of validation data is unseen for the model.
- After training the model I used validation set to get predictions and based on those predictions I have derived different types of matrices and scores.

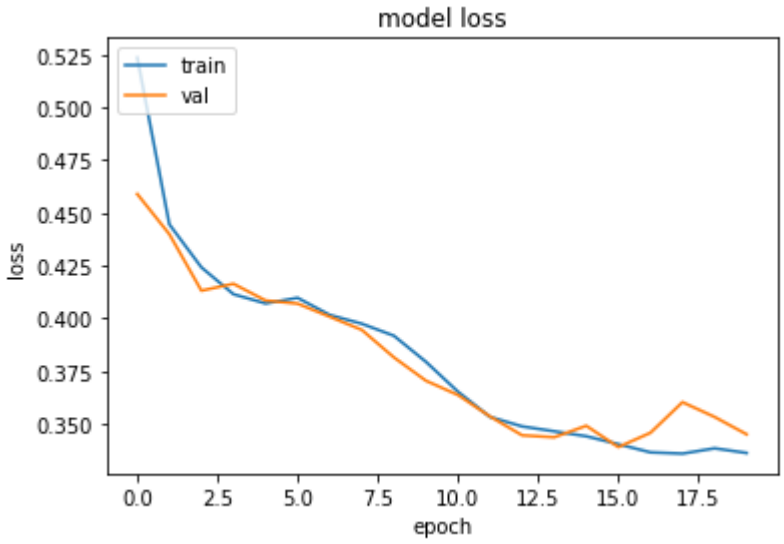
  They are presented below.

1. **Accuracy plot**



The plot shows that model achieved ~87% training accuracy and ~86% testing accuracy after 20 epochs.

2. Loss plot



The plot shows that loss has decreased gradually over the time and there are no signs of overfitting.

3. Confusion Matrix Score

```
# Making the Confusion Matrix
cm = confusion_matrix(df_test.Exited, y_pred)
print("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[1547    8]
 [ 343   92]]
```

4. Accuracy Score

```
#getting accuracy score
score = accuracy_score(df_test.Exited, y_pred)
print("Accuracy Score : \n {} %".format(int(score*100)))
```

```
Accuracy Score :
 82 %
```

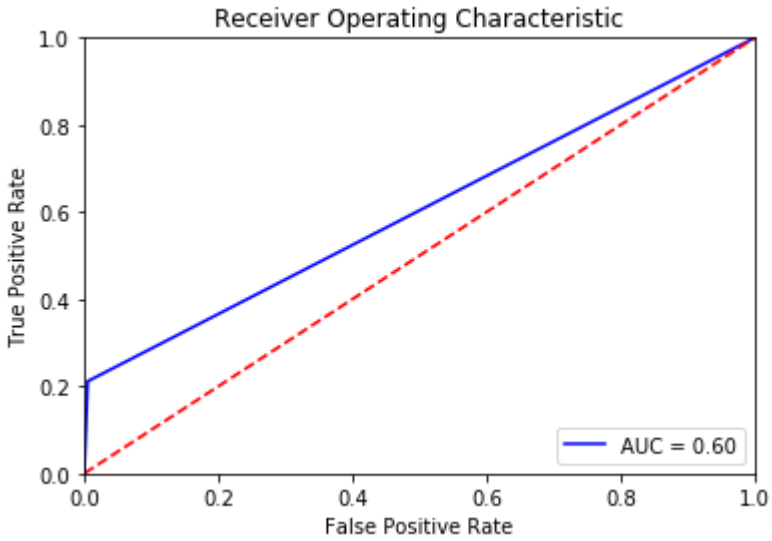
5. Classification Report – Precision, Recall and F1 Score

```
Classification Report :
              precision    recall  f1-score   support

           0       0.82      0.99      0.90      1555
           1       0.92      0.21      0.34       435

    accuracy                           0.82      1990
   macro avg       0.87      0.60      0.62      1990
weighted avg       0.84      0.82      0.78      1990
```

6. ROC-AUC Score and Curve

```
roc_auc = roc_auc_score(df_test.Exited, y_pred)
print("ROC and AUC score : \n {}".format(roc_auc))
```

```
ROC and AUC score :
 0.6031747791699006
```



4.2 Justification

- **Previous Solutions**
  1. Decision Tree Model

	precision	recall	f1-score	support
0	0.81	0.94	0.87	1300
1	0.67	0.37	0.48	458
avg / total	0.77	0.79	0.77	1758

Source: https://towardsdatascience.com/churn-prediction-770d6cb582a5

  2. Different ML models accuracy from kaggle 1

```
clf = GaussianNB()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

```
0.784
```

```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

```
0.7915
```

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy_score(pred, y_test)
```

```
/opt/conda/lib/python3.6/site-packages
433: FutureWarning: Default solver wil
pecify a solver to silence this warnin
  FutureWarning)

0.787
```

Source: https://www.kaggle.com/nasirislamsujan/bank-customer-churn-prediction/execution

3. Different ML models accuracy from kaggle 2

	name_model	accuracy_training	recall_testing	f1_score	precision_test
3	random forest	1.000000	0.549296	0.615558	0.700000
2	gradient boost	0.801510	0.756539	0.608414	0.508796
1	adaboost	0.774497	0.758551	0.596991	0.492167
0	logistic regression	0.665352	0.653924	0.421530	0.311005

Source: https://www.kaggle.com/omarzakariasalah/bank-customer-churn-modeling

- As per the statistics displayed above, let's compare the F1 scores and accuracy.

Algorithm	F1 Score	Accuracy
Random Forest	0.61	0.78
Decision Tree	0.6	0.79
Ada Boost	0.59	0.78
Logistic Regression	0.42	0.78
ANN	0.78	0.86

- This clearly shows us that my proposed solution (all the matrices are present in previous section) is outperforming all of the previous solution by high margin.
- The accuracy, precision, recall, F1 score and AUC score is way higher in ANN based approach.
- So the final model and solution is significant enough to have adequately solved the problem.

V. Conclusion

5.1 Free-form visualization

- Input data is pre-processed and normalized in such a way that it could make training significantly faster and stable.
- Here is the sample of the final **input data** for the training.

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreG
2341	0	0.734	0.094595	0.7	0.433104	0.333333	0.950092	0.000054	0.560000	0.596755
6414	0	0.536	0.040541	0.2	0.500932	0.000000	0.288780	0.000205	0.190476	0.615507
4599	0	0.582	0.270270	0.3	0.000000	0.333333	0.582329	0.000000	0.157895	0.300879
1884	0	0.612	0.229730	0.6	0.000000	0.333333	0.007369	0.000000	0.342857	0.347833
8148	0	0.860	0.391892	0.7	0.342793	0.000000	0.189826	0.000213	0.297872	0.294048

HasCrCard	IsActiveMember	Geography_France	Geography_Spain	Geography_Germany	Gender_Female	Gender_Male
1	-1	1	-1	-1	1	-1
-1	-1	1	-1	-1	1	-1
1	-1	1	-1	-1	1	-1
1	-1	1	-1	-1	-1	1
1	1	-1	1	-1	-1	1

- We already know that ANN has achieved very good accuracy on the data.
- The ANN has achieved minimum loss and highest accuracy within 20 epochs.
- Here is the screenshot of the final epoch.

```
Epoch 20/20
6400/6400 [==============================] - 1s 165us/step - loss: 0.3363 - accuracy: 0.8659 - val_loss: 0.3451 - val_accuracy:
0.8575
```

- Also we are getting prediction results based on the defined threshold.
- Threshold value is 0.5. Any value below 0.5 will be considered as false.
- Same way any value above 0.5 will be considered as true.
- The **prediction function** and the final **result** looks like this.

```
# Predicting the Test set results
y_pred = classifier.predict(df_test.loc[:, df_test.columns != 'Exited'])
y_pred = (y_pred > 0.5)
y_pred
```

```
array([[ True],
       [ True],
       [False],
       ...,
       [False],
       [False],
       [ True]])
```

5.2 Reflection

- The end-to-end solution starts with the data exploration. Where I explored data and had some dirty (not in depth) visualization.
- Then I started EDA process where I plotted important aspects and derived useful insights.
- Next I performed feature engineering in order to get important features and discard irrelevant features.
- Afterwards, I designed neural network architecture and tuned hyper-parameters to get optimal set of weighs.
- And finally, I plotted some interesting evaluation matrices.
- The important and most difficult aspect of the project is data imbalance in dataset. This makes this problem more complex.
- However we managed to get final robust solution for this problem. This solution can help organizations to predict the churning rate and help to take precautions and prevention.

5.3 Improvement

- We can make more improvements on this ANN based approach because it's all about hyper-parameters. One can easily manipulate with them and make significant boots in the existing accuracy.
- In my opinion I would play with model depth (hidden layers and number of neurons) to achieve more accuracy.
- There is an existing solution based on XGBoost algorithm which has a little higher accuracy than our solution. However unlike our solution, XGBoost based solution sometimes fails to generalize well on new data.

References

[1] https://www.kaggle.com/adammaus/predicting-churn-for-bank-customers

[2] https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9?gi=27bc4a183c41

[3] https://www.researchgate.net/publication/251626579_Introduction_to_Artificial_Neural_Network_ANN_Methods_What_They_Are_and_How_to_Use_Them

[4] https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a

[5] https://www.researchgate.net/publication/236625937_A_Proposed_Churn_Prediction_Model

[6] https://towardsdatascience.com/churn-prediction-770d6cb582a5

[7] https://www.researchgate.net/publication/310757545_A_Survey_on_Customer_Churn_Prediction_using_Machine_Learning_Techniques

[8] https://www.kaggle.com/nasirislamsujan/bank-customer-churn-prediction/execution

[9] https://www.kaggle.com/omarzakariasalah/bank-customer-churn-modeling