CUSTOMER CHURN PREDICTION

CS 5661 – DATA SCIENCE

PROJECT REPORT

Team Members:

1. Mujahir Hussain Abbasi
2. Rutwik Vilas Wagh
3. Shivam Sanjay Jadhav
4. Abir Sur
5. Harshit Babu
6. Abhishek Singh Rajput
7. Akash Chavan
8. Zeeshaan Uddin Mohammed

Responsibilities:

1. Mujahir Hussain Abbasi:
   - Exploratory Data Analysis
   - Researched and Implemented XGBoost

2. Rutwik Vilas Wagh
   - Researched and implemented ANN
   - Contributed in the project report

3. Shivam Sanjay Jadav:
   - Model Performance Evaluation
   - Contributed in the project report

4. Abir Sur:
   - Data Visualization
   - Contributed in the Power Point Presentation

5. Harshit Babu:
   - Data Preprocessing
   - Contributed in the PowerPoint Presentation

6. Abhishek Singh Rajput:
   - Data Visualization
   - Contributed in the PowerPoint presentation.

7. Akash Chavan:
   - Data Down Sampling
   - Contributed in the PowerPoint Presentation

8. Zeeshaan Uddin Mohammed:
   - Researched and Implemented ANN.
   - Contributed in the project report

Abstract:

The customer churn prediction ML project aims to develop a predictive model that can identify customers who are likely to leave a company's services or products. The model utilizes various machine learning algorithms and techniques to analyze customer behavior, past interactions, and demographic data to identify patterns that indicate potential churn. The end result is a tool that can help companies anticipate customer churn and take proactive steps to retain valuable customers.

Dataset:

We have taken the dataset from Kaggle. Link

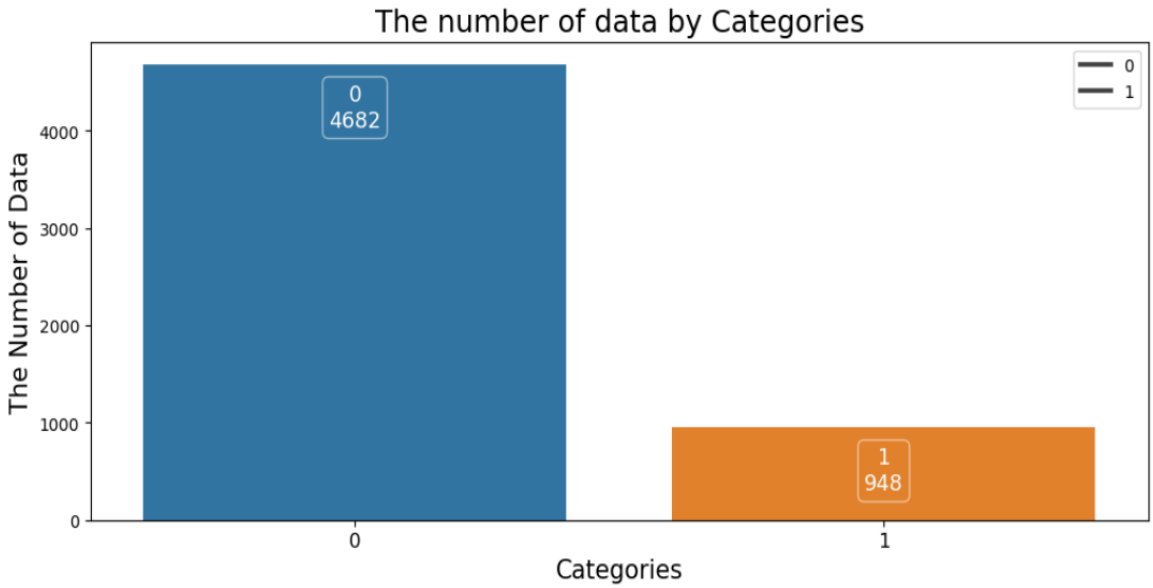
The dataset contains 2 classes. These classes represent different churn flags. The 2 classes are as follow:

- 0 – Represents non churn
- 1 – Represents churn

The dataset consists of a total of 5630 datapoints.

The data is distributed amongst 2 classes as mentioned below:

Class	Datapoints
Churn	4862
Non-churn	948



Project Description:

The customer churn prediction ML project involves developing a machine learning model that can predict which customers are likely to stop using a company's products or services. The goal of the project is to help companies retain their valuable customers by identifying those who may be at risk of leaving and taking proactive steps to address their concerns.

The project typically involves collecting and analyzing large amounts of customer data, including past behavior, demographic information, and other relevant factors. This data is used to train the machine learning model, which then generates predictions about which customers are most likely to churn.

To build an effective customer churn prediction model, various machine learning algorithms and techniques can be used, such as XGBoost and Artifictal neural network. These techniques can help identify patterns in the data that may not be immediately apparent to humans, allowing for more accurate predictions.

Once the model is trained and validated, it can be deployed as a tool for businesses to identify customers who are at risk of leaving. Companies can then take proactive steps to retain these customers, such as offering incentives, improving customer service, or addressing any concerns or complaints they may have.

Overall, the customer churn prediction ML project is a valuable tool for companies looking to improve customer retention and reduce churn rates. By leveraging the power of machine learning, businesses can gain a deeper understanding of their customers and take action to keep them satisfied and engaged.

Project Goals:

The primary goal of the customer churn prediction ML project is to develop a predictive model that can accurately identify customers who are likely to stop using a company's products or services. By doing so, the project aims to help businesses reduce customer churn rates and improve customer retention.

The specific objectives of the project may include:

1. Collecting and analyzing customer data to identify patterns and trends that may indicate potential churn.
2. Selecting and implementing appropriate machine learning algorithms and techniques to build a predictive model that can accurately identify at-risk customers.
3. Validating and testing the model to ensure it provides accurate predictions.
4. Deploying the model as a tool for businesses to identify at-risk customers and take proactive steps to retain them.
5. Monitoring the model's performance over time and making adjustments as needed to maintain its accuracy and effectiveness.

Overall, the goal of the customer churn prediction ML project is to help businesses improve customer satisfaction, reduce churn rates, and increase revenue by retaining valuable customers.

Objectives:

The goal is to predict the customers who are likely to churn or stopped purchasing your business's products or services during a certain period of time.

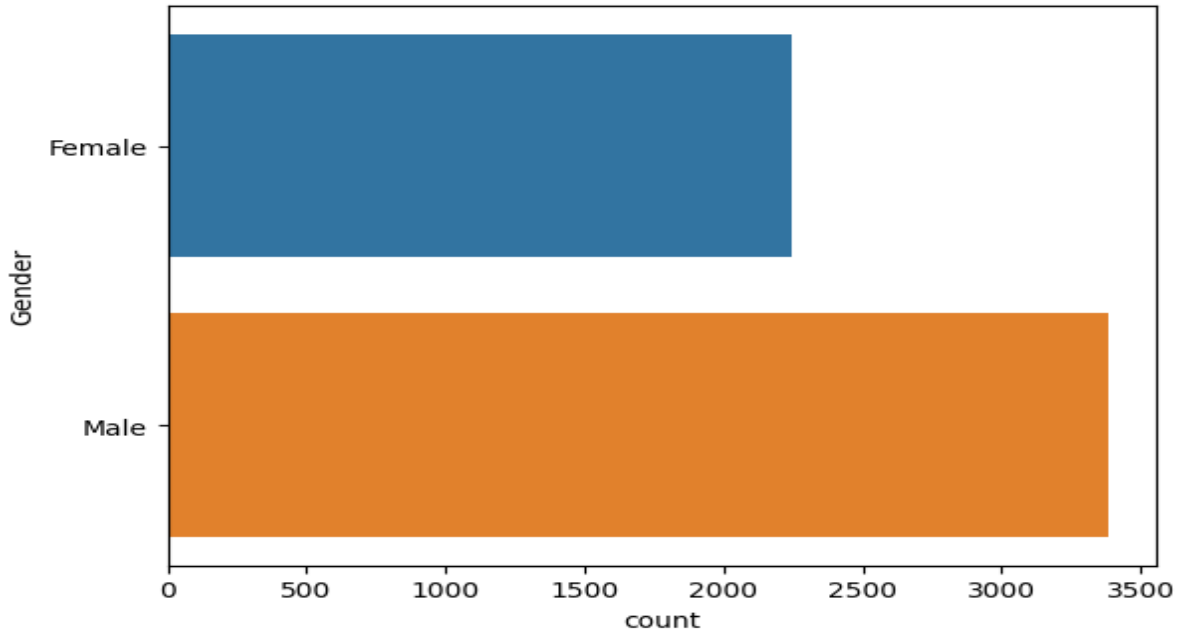
Technologies and Libraries:

- Programming Langauge : Python
- Technology : Machine Learning
- Platform : Google Colab
- Code Editor : VScode, Jupyter Notebook
- Libraries : Numpy, Pandas, Sklearn, Tensorflow, Keras, Matplotlib, Seaborn

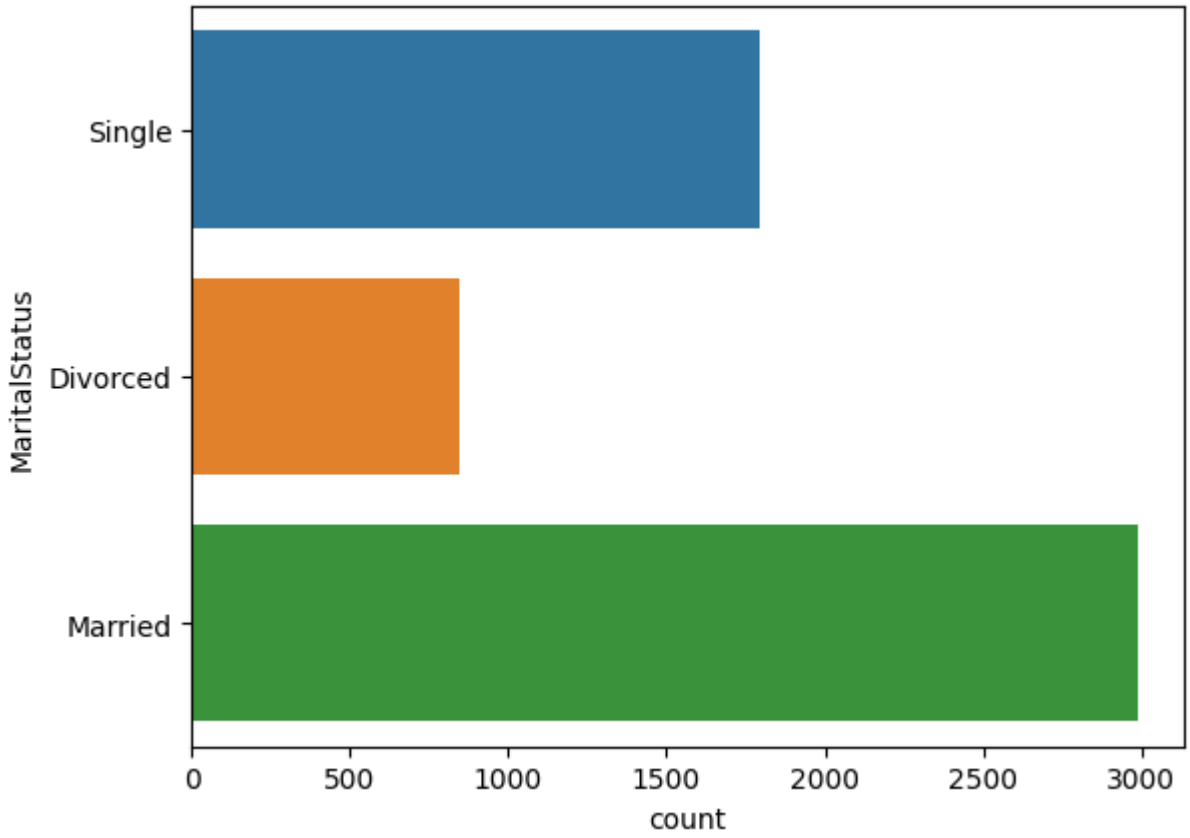
Exploratory Data Analysis:

1. Data Analysis:

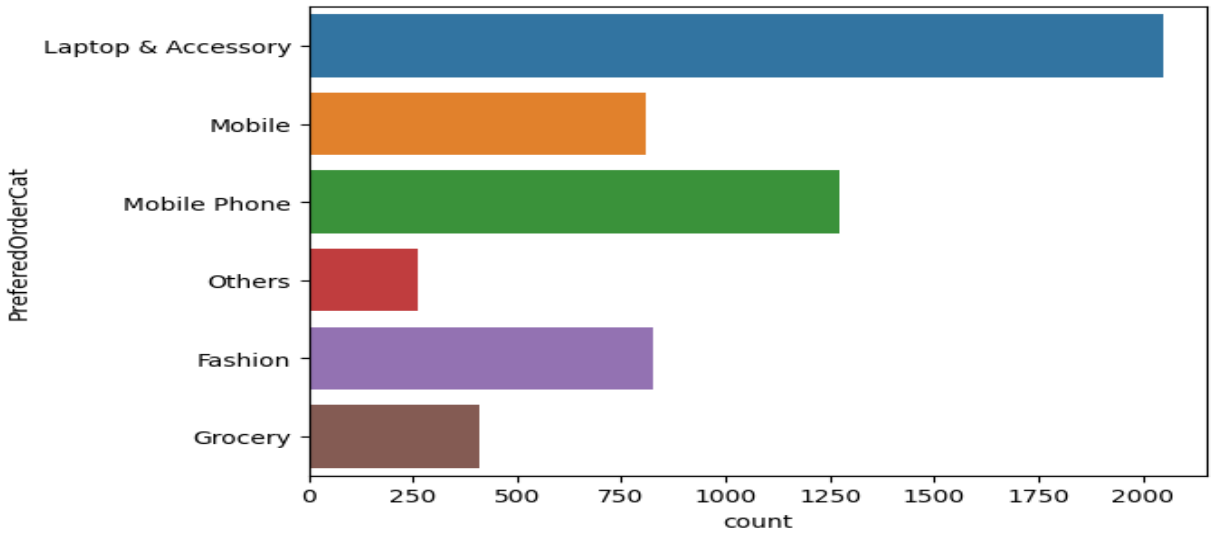
- Out of 5630 rows, Missing values are 1856, So Nearly 30% data has missing values.
- Out of 5630 rows, 4682 represents the non-churn customers and 928 churn customers, So it clearly shows that the data is imbalanced.
- The Customer ID has Unique values, So it is a redundant column
- Most of the customers in the dataset are male



- Most of the customers are married followed by single and divorced.

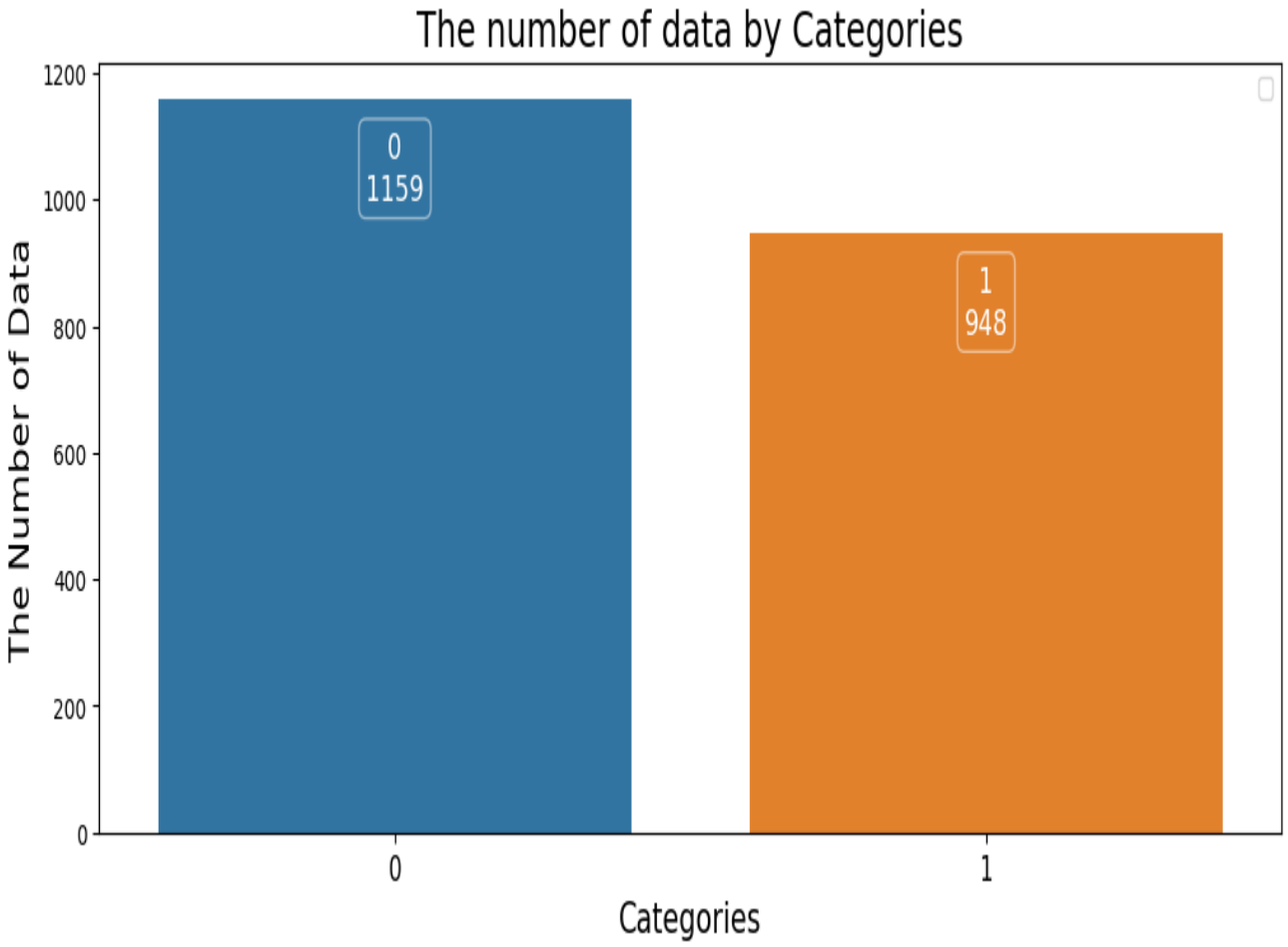


- Laptops and mobile phones are the most ordered category among the customers



2. Data Preprocessing:

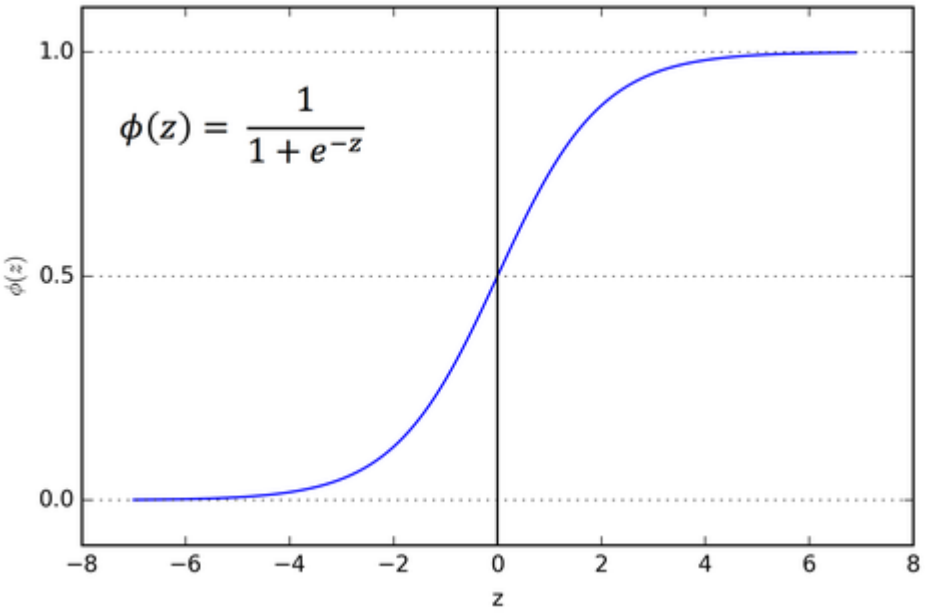
- Missing values are handled by the XGBoost algorithm itself while training. But, while training the ANN model, The missing values have been imputed by the median of the specific column
- Categorical column GENDER has been label encoded by considering that each gender has some impact on the result. While all other categorical columns which have more than two values has been one-hot encoded.
- Imbalanced Data issue has been addressed by implementing the down sampling technique which is reducing the non-churn customer data to 25%. Down sampled dataset now has 1159 non-churn customers and 948 churn customers.



Activation Functions:

1. Sigmoid Activation Function:

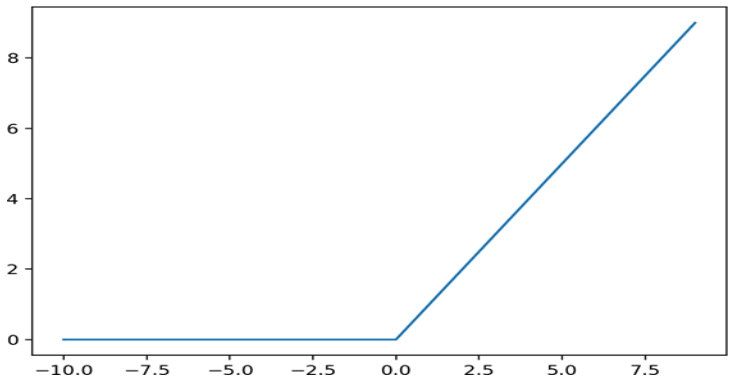
The sigmoid activation function is a mathematical formula commonly used in neural networks to introduce nonlinearity into the model's output. It transforms any input value into a range between 0 and 1, which makes it useful for binary classification problems where the output is either 0 or 1. The function is defined as 1 / (1 + e^-x), where x is the input value. As x approaches positive infinity, the output of the function approaches 1, and as x approaches negative infinity, the output approaches 0. However, for input values near 0, the function produces a smooth curve that gradually transitions between the two extremes.



2. ReLU Activation Function:

ReLU stands for the Rectified linear unit. It is a linear function that outputs the Input directly if it is positive or else it outputs zero.

a = max(0,z)



Algorithms:

1. XGBoost :

XGBoost (eXtreme Gradient Boosting) is a popular machine learning algorithm that is widely used for classification and regression tasks. It is an ensemble method that combines the predictions of many weak learners (usually decision trees) to produce a more accurate final prediction.

The XGBoost algorithm works by iteratively training new decision trees to correct the errors made by the previous trees. Each new tree is trained to predict the residuals (the difference between the actual target values and the predicted values) of the previous trees. This allows the model to gradually improve its predictions with each new iteration.

XGBoost is known for its speed and accuracy, and it has been used to win numerous data science competitions. It also has many hyperparameters that can be tuned to improve its performance, such as the learning rate, maximum depth of the trees, and the regularization parameter. However, tuning these hyperparameters requires careful experimentation and can be time-consuming.

2. Artificial neural networks:

Artificial neural networks (ANNs) are a type of machine learning algorithm that are modeled after the structure and function of the human brain. They consist of layers of interconnected nodes (also known as neurons) that process information and make predictions based on the input data.

ANNs are typically trained using a process known as backpropagation, which involves adjusting the weights and biases of the neurons to minimize the difference between the predicted output and the actual output. This process is repeated many times on a large dataset to improve the accuracy of the model.

ANNs can be used for a wide range of tasks, including image classification, speech recognition, and natural language processing. They are particularly useful for tasks that involve large amounts of complex data, as they can learn to extract important features from the data and make accurate predictions based on those features. However, ANNs can be computationally expensive and time-consuming.

Implementation and Code:

XGBoost:

Step wise Implementation

- Training an XGBoost model involved several steps. First, the data is preprocessed and split into training and validation sets. Next, the model hyperparameters have been defined, such as the learning rate, number of trees, and max depth.

- Since the dataset is small, the cross validation score was used to evaluate the consistency of the model. The model is then trained on the training set using boosting technique, which involves iteratively adding decision trees to minimize the loss function.

- During each iteration, the model evaluates the performance on the validation set to determine if further training is necessary or if the model is overfitting. Once the model has converged, It is then used to make predictions on new data. Model performance has been evaluated using metrics such as accuracy, precision, recall, and F1 score.

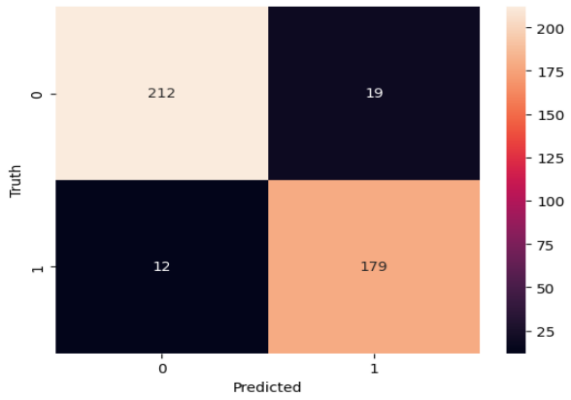
Fitting the model:

```
In [88]: clf_xgb = xgb.XGBClassifier()
         grid_values = {'max_depth': [3,6,20],'n_estimators':[100,500,1000],'learning_rate':[0.01,0.1,0.3]}
         grid_clf_xgb = GridSearchCV(clf_xgb, param_grid = grid_values, cv = 5)
         grid_clf_xgb.fit(X_train, Y_train)

Out[88]: GridSearchCV(cv=5,
                      estimator=XGBClassifier(base_score=None, booster=None,
                                              callbacks=None, colsample_bylevel=None,
                                              colsample_bynode=None,
                                              colsample_bytree=None,
                                              early_stopping_rounds=None,
                                              enable_categorical=False, eval_metric=None,
                                              gamma=None, gpu_id=None, grow_policy=None,
                                              importance_type=None,
                                              interaction_constraints=None,
                                              learning_rate=None, max_bin=None,
                                              max_cat_to_onehot=None,
                                              max_delta_step=None, max_depth=None,
                                              max_leaves=None, min_child_weight=None,
                                              missing=nan, monotone_constraints=None,
                                              n_estimators=100, n_jobs=None,
                                              num_parallel_tree=None, predictor=None,
                                              random_state=None, reg_alpha=None,
                                              reg_lambda=None, ...),
                      param_grid={'learning_rate': [0.01, 0.1, 0.3],
                                  'max_depth': [3, 6, 20],
                                  'n_estimators': [100, 500, 1000]})
```

Model Accuracy Report and Confusion Matrix:

```
In [95]: #Molde Evaluation metrivs
         cm = confusion_matrix(y_test,pred)
         print(classification_report(y_test,pred))

                       precision    recall  f1-score   support

                    0       0.95      0.92      0.93       231
                    1       0.90      0.94      0.92       191

             accuracy                           0.93       422
            macro avg       0.93      0.93      0.93       422
         weighted avg       0.93      0.93      0.93       422
```

```
In [96]: sn.heatmap(cm, annot=True, fmt='d')
         plt.xlabel('Predicted')
         plt.ylabel('Truth')
Out[96]: Text(50.722222222222214, 0.5, 'Truth')
```



Artificial Neural Network:

Step wise Implementation

- There are four hidden layers with the activation function as ReLu and Sigmoid
- The Binary cross entropy loss function has been used while training, Since it is a binary class problem
- Adam optimizer has been used in this model
- The model is trained for 150 Epochs

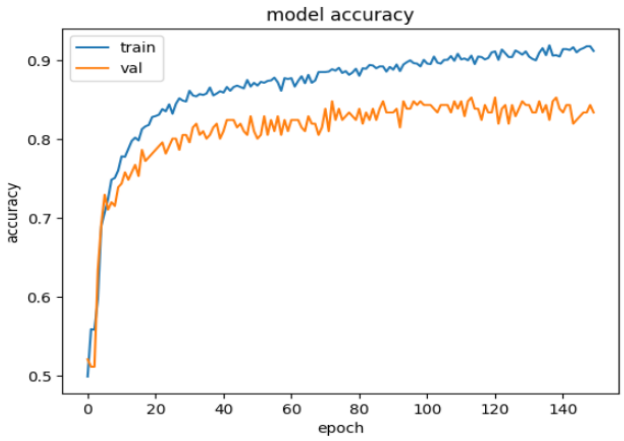
Model Building:

```
In [22]: model = keras.Sequential([
             keras.layers.Dense(20, input_shape=(33,), activation='relu'),
             keras.layers.Dense(15, activation='relu'),
             keras.layers.Dense(5, activation='relu'),
             keras.layers.Dense(1, activation='sigmoid')
         ])


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

         history = model.fit(X_train, y_train, epochs=150, validation_data=(X_valid, y_valid))
```

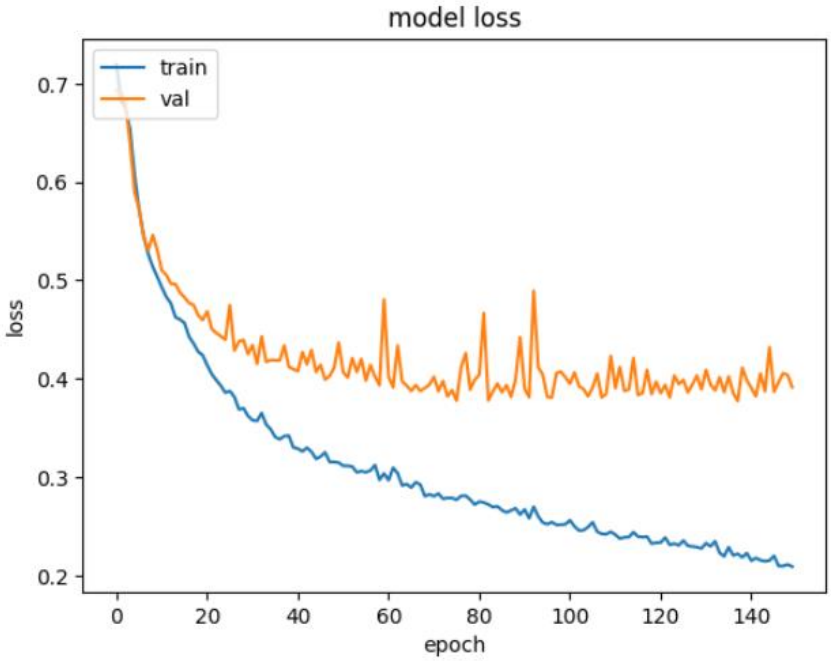
```
Epoch 1/150
53/53 [==============================] - 12s 17ms/step - loss: 0.7197 - accuracy: 0.4994 - val_loss: 0.6928 - val_accuracy: 0.5
213
Epoch 2/150
53/53 [==============================] - 1s 10ms/step - loss: 0.6829 - accuracy: 0.5594 - val_loss: 0.6914 - val_accuracy: 0.51
18
Epoch 3/150
53/53 [==============================] - 0s 8ms/step - loss: 0.6750 - accuracy: 0.5588 - val_loss: 0.6799 - val_accuracy: 0.511
8
Epoch 4/150
53/53 [==============================] - 0s 8ms/step - loss: 0.6531 - accuracy: 0.5974 - val_loss: 0.6388 - val_accuracy: 0.635
1
Epoch 5/150
53/53 [==============================] - 1s 11ms/step - loss: 0.6066 - accuracy: 0.6900 - val_loss: 0.5891 - val_accuracy: 0.69
19
Epoch 6/150
53/53 [==============================] - 1s 10ms/step - loss: 0.5713 - accuracy: 0.7072 - val_loss: 0.5724 - val_accuracy: 0.72
99
Epoch 7/150
53/53 [==============================] - 1s 11ms/step - loss: 0.5452 - accuracy: 0.7245 - val_loss: 0.5433 - val_accuracy: 0.71
09
Epoch 8/150
```

Training and Validation Accuracy:

```
In [23]: plt.plot(history.history['accuracy'])
         plt.plot(history.history['val_accuracy'])
         plt.title('model accuracy')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'val'], loc='upper left')
         plt.show()
```



Training and Validation Loss:



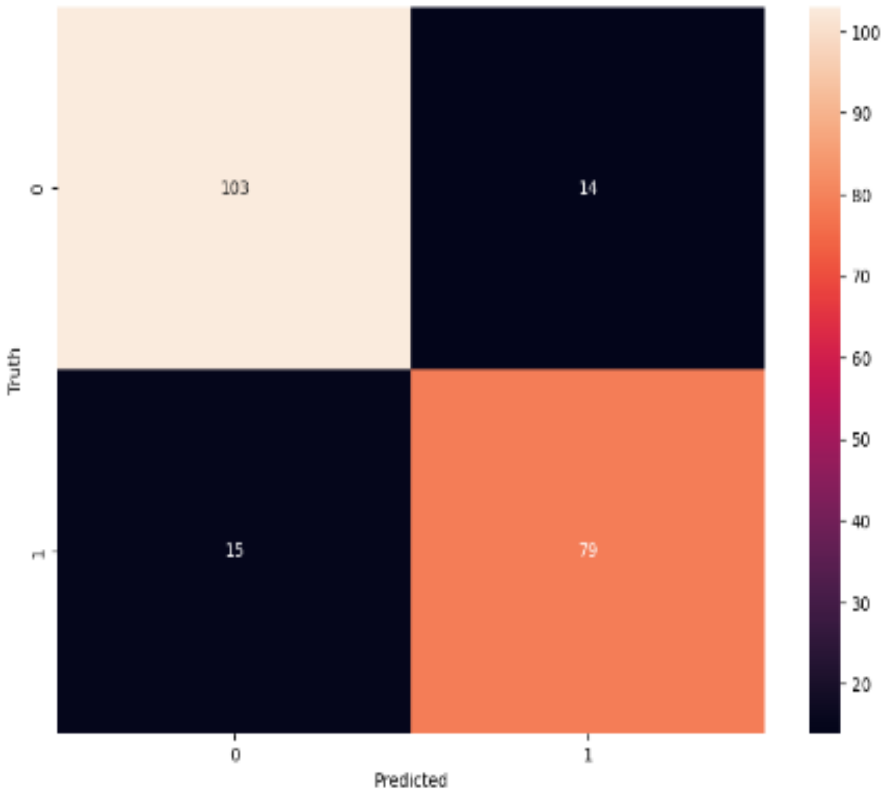
Model Accuracy Report and Confusion Matrix:

In [28]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.88      0.88       117
           1       0.85      0.84      0.84        94

    accuracy                           0.86       211
   macro avg       0.86      0.86      0.86       211
weighted avg       0.86      0.86      0.86       211
```

In [29]:
```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[29]: Text(95.72222222222221, 0.5, 'Truth')



Accuracy Comparison:

		XGBoost	ANN
0 (Non-churn)	Precision	0.95	0.87
	Recall	0.92	0.91
	F1 score	0.93	0.89
	accuracy	0.93	0.88
1 (Churn)	Precision	0.90	0.89
	Recall	0.94	0.84
	F1 score	0.92	0.87
	accuracy	0.93	0.88

XGBoost and Artificial Neural Network have been trained to implement the Customer Churn Prediction. The original imbalanced data has been down sampled to make it a more realistic and reliable data and ANN model achieved the 86% accuracy whereas the XGBoost model has achieved the highest accuracy of 93% percent with equal percentage of F1- Score.