CUSTOMER CHURN PREDICITON

Sachin Gupta(Enrollment No.- 22125031) Abhi(Enrollment No.- 22125002)

Abstract—In this project, we are going to predict whether a regular customer of a company is going to churn or stop using the product of that company or not. So, our main goal is to suggest the optimal machine learning algorithm for early client churn prediction. We have used a technique to generate synthetic data in which we are increasing the number of samples of minority classes to make the data balanced and checking the accuracy for both the data set with and without SMOTE. Then, We have done the hyper-parameter tuning using Bayesian optimization to find out which hyper-parameters are giving us the best results.Then, We are comparing the results for all models which we have taken and finding out the best model which can be used for customer churn prediction. In this project we have tested algorithms like Logistic Regression, KNN, Neural Network, Decision Tree, Random Forest, Gradient Boosting Classifier, AdaBoost, XGBoost.

I. INTRODUCTION

Churning refers to the number of customers who stopped using a particular product. Every company wants that there customers' churning rate must be low.The reason behind the churning is that when the customer has got same type of product from other company with minimum cost as well as best quality. For company retaining existing customers is equally important as gathering new customers.

Generally customers will churn and opt for other company when they face the issues of high cost or low quality of product as compared to other company. If businesses can predict the customers who are going to churn then can segment those customers and will try to provide them better services to retain them. Hence, the churn prediction model has become compulsory for businesses in today's digitized economy. An organization can achieve a high customer retention rate and maximize its revenue.

If we can identify the possible churn early then the company can take the proactive measures to retain that customer and it will help in preventing revenue loss. Companies need to compete to attract new customers as well as to retain the existing ones.

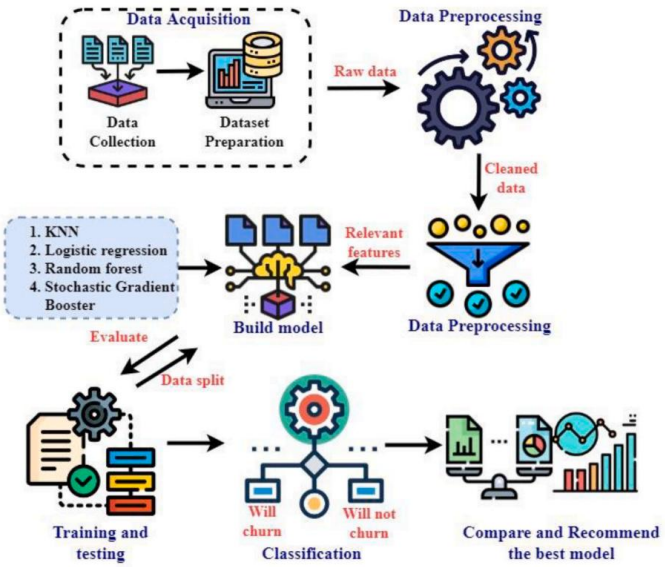
Some of the main churn prediction algorithms which can identify the most necessary variables that are affecting the target variable are utilized in this project: Logistic Regression, KNN, Neural Network, Decision Tree, Random Forest, Gradient Boosting Classifier, AdaBoost, XGBoost.

II. STUDY FLOW

Customer churn prediction has been performed using different methods, techniques, data processing, machine learning. Most of these methods were using algorithms of trees.As we know that decision trees over-fit the data so reducing the information improves the accuracy of the decision trees.

The data set which we have taken is from the Kaggle with 7043 alternatives and 21 attributes was taken as input. Initially, to convert the categorical data into numerical data we have used one hot encoding as well as the label encoding and for the numerical attributes we have used knowledge MinMaxScaler to make range of all attributes [0,1]. Then, the information has been parted into two sections, train,and test set separately.

Since our data set is imbalanced i.e. number of persons churning are 1869 and number of persons not churning are 5174. So, I have used SMOTE.



III. Methods Used

The system involved in the analysis of customer churning uses different algorithms mentioned below:

(i) Logistic regression: It is a supervised learning method which is widely used for binary classification. The output predicts the probability that a given input belongs to one of the two predefined category. It uses the sigmoid or logistic function to transform the linear combination of the features into the probability values ranging from [0,1].
The threshold is chosen which is usually 0.5. If the predicted probability is above this threshold, the input is classified into one category; otherwise, it's classified into the other.

The probability that the input belongs to a category is:

$$P(x) = \frac{1}{e^{-(\beta 0 + x * \beta 1)}}$$

Here the loss function is the cross entropy. Log loss for the k^{th} point is :

$$L_k = - y_k * \ln(p_k) - (1 - y_k) * \ln(1 - p_k)$$

For the parameter estimation minimizing the Loss function w.r.t to β_0 and β_1:

$$\frac{\partial L}{\partial \beta_0} = 0, \quad \frac{\partial L}{\partial \beta_1} = 0$$

Logistic regression is widely used because it's simple, efficient, and interpretable. It's particularly useful when the relationship between the independent variables and the dependent variable is linear, and the dependent variable is binary.

(ii) Neural Network: In the Neural Network we have the input then there are the hidden layers and then there is the output.It is just like the human brain where neurons are intercoonected and arranged in layers.

Input is multiplied by the corresponding weights of the hidden layer and then the activation function is applied which can be sigmoid, ReLU, tanh, softmax.The activation functions helps in capturing the non linearity in the data. Also they must be monotonic in nature otherwise they will create problem of local minima or maxima.

Other problem which arise in this is vanishing gradient descent or exploding gradient descent.For minimizing the loss function we have to update the weights and the biases which is done using gradient descent in which we define a learning parameter .

It is should be adjusted such that it neither very small otherwise it will be very computationally complex nor very large such that it skip the maxima or minima.
For faster updates we can also use the momentum
So that if we are going down the slope again and again then our updates values increases faster and we can reach the minima easily.

Since there are many different types of activation functions so it can also capture the complex relationship which is making it helpful for classification, regression, pattern recognition, etc.

(iii)KNN: K-Nearest Neighbors (KNN) is an example of a instance-based learning algorithm (i.e., it remembers training data rather than learning specific patterns) in which we are taking the k nearest neighbour for the data-point and taking the majority of the classes as its final category.

For the regression tasks we take the average of all the values of k-nearest neighbour and update it as the final answer for the new data-point. To find the nearest neighbors we can use the measurements like Euclidean

distance or manhattan distance.

We can also take all the neighbors but along with their weights which is inversely proportional to he euclidean distance or manhattan distance between them.

Let $x_1 \dots x_k$ denote the k instances from training examples that are nearest to x_q.

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

$$\text{where } \delta(a,b) = 1 \quad \text{if } a = b$$
$$= 0 \quad \text{otherwise}$$

The inductive bias of KNN is that all the data-points which are of same category belongs to the same cluster or the are near as compare to the data-points which are of different category.

(iv) Decision Tree: Decision tree is the supervised machine learning algorithm in which we classify the data-points by traversing the tree and comparing the features by their nodes values. It can be used for both classification as well as regression tasks.

For doing the classification tasks data-set is splitted by the decision tree into the subsets based on the most significant attributes/ features which is done using information gain,gain ration or gini index. It continuously divides the data set into smaller and smaller subsets until the goal of maximizing the purity of the resulting subset is reached.

For the numerical attributes, data-set is sorted according to the features and a threshold is decided. Then we can decide its category by the threshold value.

For classifying a data-point to know which category it belongs to we start from root node and check the condition written on the node according to which we move forward. Each node classifies the data and there are many option below it and we have to take the correct path. When we reaches the leaf node the final category written on it will be our answer.

For deciding the feature on which we have to split the data we can use different types of algorithms like Gini Impurity, Information Gain (Entropy), and Gain Ratio. They measure the percentage of impurity of uncertainty in the data i.e. if we have 100 points of one category and 0 point of other category then it is very easy to classify them and our impurity is lowest but if there are 50 point of one category and 50 point of other category then the impurity is highest.

$$\text{Entropy}(S) = -\sum_{1}^{c} p_i * \log(p_i)$$

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

If S is divided into 3 subsets then :

$$gini_A(S) = \frac{|S_1|}{|S|} gini(S_1) + \frac{|S_2|}{|S|} gini(S_2) + \frac{|S_3|}{|S|} gini(S_3)$$

But there is one major issue in random forest which is over-fitting. Since we can splits the nodes again and again we can finally reach the leaf nodes where all the data-points can be classified correctly. But if our data is noisy or outliers is present in our data then it will give zero error for training data but very high error on testing data i.e. it will not generalize the data.

To stop over-fitting: We can stop the tree to grow it to the point where it perfectly classifies the data which can be done using early stopping or by deciding the maximum depth or minimum points in the leaf nodes. Other way is to allow the tree to over-fit then post prune the tree.

(v) Random Forest: Due to the issue of over-fitting issue of the decision tree as well as giving the high variance ensemble method was developed in which we are taking so many decision trees and also sampling the instances to pass different data-points to different trees.

It offers two sources of randomness: “bagging” and “random input vectors”.

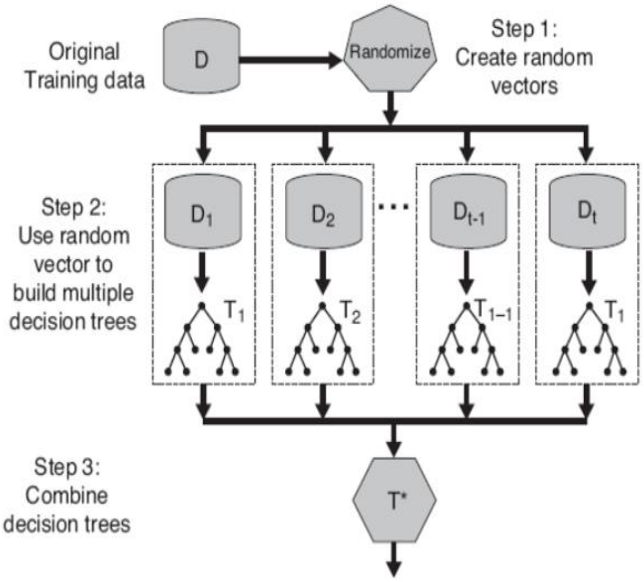
Bagging method: Due to bagging the instances are sampled without replacement and there is probability of 63.2% that instances have been taken in at-least one decision tree. Since we are using many different decision trees so there is possibility that the variance will decrease and generalization improves.

Random Input vector: To create more randomness features are also sub sampled and at each node we will

take randomly ‘m’ features and split the node according to the best of the ‘m’ available nodes. So in each part of the decision trees, a random subset of features is chosen instead of using all features which further eliminate the possibility of trees correlation and diversify the algorithm.

For classification the instance, final prediction of all the trees are taken and the category which majority of the trees get will be our final answer. For regression tasks, this is the average estimate of all individual trees.

If we compared random forest with the decision tree then it is flexible and gives better results and does generalization well and also does not affect highly by noisy data and outliers.



(vi) Gradient Boosting Classifier: It is an machine learning algorithm which is based on boosting in which there are multiple weak learners or decision stumps which combines to predict the final result. First decision tree only contains the average of all the target values. Then the second tree trains on the residuals i.e. the difference between the actual value and the predicted value by the first tree. Similarly, other trees will train on the residual from the all previous trees.

Here, a hyper-parameter learning rate is defined. While giving the prediction it will take complete output of first tree but it will take learning rate times the output of other trees. Here regularization techniques are used to reduce the over-fitting by constraining the training procedure.

One way is to reduce the number of boosting iterations because increasing the number of iterations will result in the over-fitting of data. An optimal iterations can be selected by monitoring the prediction error on validation data-set.

Other way is to penalize the model on the complex structures i.e. L_1 or L_2 penalty can be applied on the number of leaf nodes so the model will stop early and will not over-fit the data.

The main disadvantage is that we have to train the model on the thousands of decision trees. So its implementation is more difficult due to high computational demand.

(vii) AdaBoost (Adaptive Boosting) Classifier: It is an ensemble learning method which is based on boosting that combines the predictions of many weak learners to create a strong classifier. A weak learner is a classifier that performs poorly but it is better than random guessing.

Initially every instance will have the same weights. After every iteration it will increase the weights of the all the instances which are classified incorrectly and increase the weights of correctly classified instances. Thus the model will try to improve on every instance.

For all the weak learners their weights is decided by the error of the corresponding learner. If the error is high then its weight will be less and vice-versa. Finally prediction is done by adding the results of all the weak learners and multiplying them by corresponding weights.

The formula to fins the weights as well the final prediction is given below :

$\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ where ε_t is the error of t^{th} classifer

$$H(x) = \text{sign}\left(\sum_{1}^{T} \alpha_t h_t(x) \right)$$

where α_t is the weight of t^{th} classifier

To decrease the over-fitting and improving the generalization we can use the early stopping or or the pruning in which we can those classifiers which are performing poorly and giving very high error. If two weak classifier are giving are producing similar outputs

then we can remove one of the classifier and increase the coefficient of other classifier.

Adaboost with decision trees as the weak learners is often called as best out-of-the-box classifier.

(viii) XGBoost (Extreme Gradient Boosting): XGBoost is a gradient boosting open-source software library which is designed to have all the features of other algorithms. It is one of the most popular and widely used technology libraries.

It is designed for higher speed and better performance. It includes Lasso and Ridge regression i.e. L1 and L2 penalty to improve the generalization and to prevent over-fitting. It also used the parallel processing to train the code which it makes it very efficient and also making it faster than other gradient boosting libraries.

To reduce the over-fitting it uses technique called "pruning" in which it removes those segment of the trees which are not matching performance of the model.

It has another important that it can deal with the missing values in the data-set. It also includes the cross-validation function to find the best training data and testing data and also helps in optimizing the model.

Since it is very vast library which includes mostly all different types of algorithm so it provides outstanding results. It also provides several parameters and options that can be adjusted and customized by the user according to the specific needs of the problem they are working on.

IV. RESULTS

Without hyper-parameter tuning :

Before SMOTE, different types of metrics for all the models are shown :

model	accuracy	precision	recall	f1_score
Logistic Regression	0.795571	0.650000	0.521097	0.578454
KNN	0.751846	0.542923	0.493671	0.517127
Neural Network	0.772288	0.582766	0.542194	0.561749
Decision Tree	0.738217	0.513570	0.518987	0.516264
Random Forest	0.782510	0.625344	0.478903	0.542413
Gradient Boosting Classifier	0.801817	0.663185	0.535865	0.592765
AdaBoost Classifier	0.805224	0.669251	0.546414	0.601626
XGBoost	0.776831	0.601504	0.506329	0.549828

After SMOTE, different types of metrics for all the models are shown :

model	accuracy	precision	recall	f1_score
Logistic Regression	0.759903	0.718222	0.817814	0.764789
KNN	0.779227	0.714980	0.893725	0.794422
Neural Network	0.829952	0.796089	0.865385	0.829292
Decision Tree	0.765217	0.743217	0.776316	0.759406
Random Forest	0.838647	0.815029	0.856275	0.835143
Gradient Boosting Classifier	0.822705	0.777480	0.880567	0.825819
AdaBoost Classifier	0.787440	0.740351	0.854251	0.793233
XGBoost	0.836715	0.811900	0.856275	0.833498

After hyper-parameter tuning Bayesian Optimization:

Before SMOTE, different types of metrics for all the models are shown :

model	accuracy	precision	recall	f1_score
Logistic Regression	0.800114	0.656410	0.540084	0.592593
Gradient Boosting Classifier	0.801249	0.663158	0.531646	0.590164
AdaBoost Classifier	0.804656	0.669271	0.542194	0.599068
XGBoost	0.804656	0.673797	0.531646	0.594340
Random Forest	0.801817	0.687688	0.483122	0.567534
KNN	0.780806	0.612821	0.504219	0.553241
Neural Network	0.801817	0.672176	0.514768	0.583035
Decision Tree	0.783078	0.648387	0.424051	0.512755

After SMOTE, different types of metrics for all the models are shown :

model	accuracy	precision	recall	f1_score
Logistic Regression	0.761836	0.719610	0.820850	0.766903
Gradient Boosting Classifier	0.854589	0.837095	0.863360	0.850025
AdaBoost Classifier	0.800483	0.753304	0.865385	0.805464
XGBoost	0.841546	0.808989	0.874494	0.840467
Random Forest	0.837681	0.802412	0.875506	0.837367
KNN	0.823671	0.772528	0.893725	0.828719
Neural Network	0.840580	0.801282	0.885628	0.841346
Decision Tree	0.791787	0.755270	0.834008	0.792689

As we can see from the tables that after SMOTE accuracy, precision, recall all have improved because before SMOTE the results were bias towards churn = YES category but when both the categories got same proportion it leads to learn from minority class more effectively and hence accuracy improves.

Figure below is the confusion matrix before SMOTE :

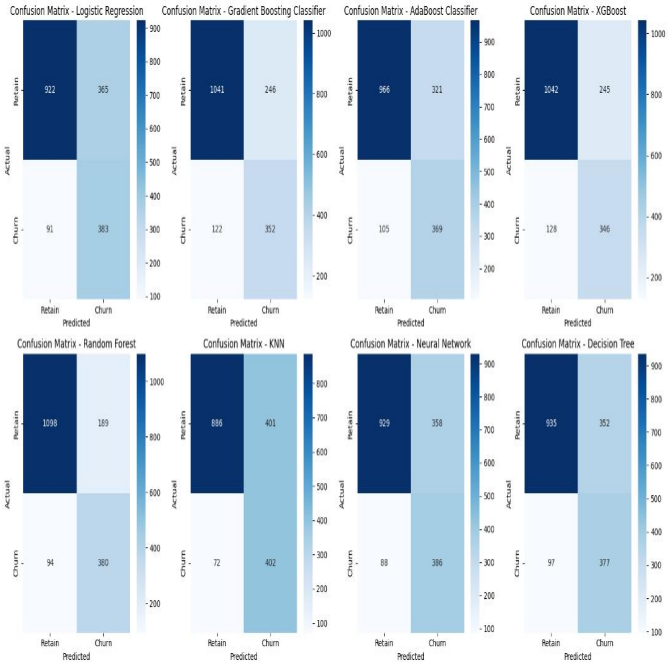


Figure below is the confusion matrix after SMOTE :

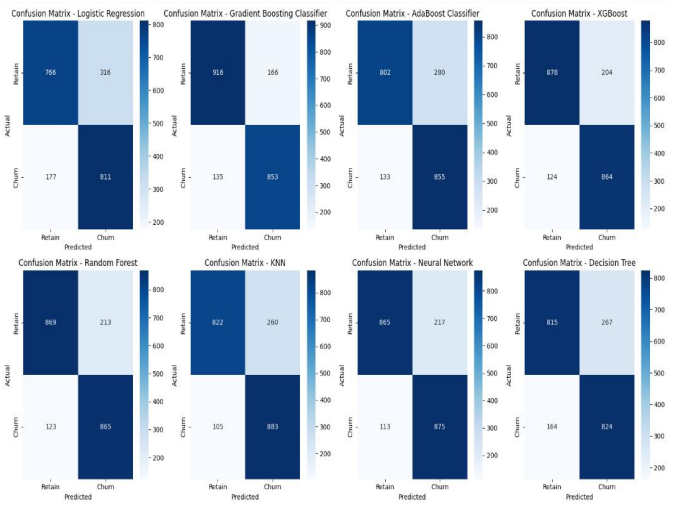
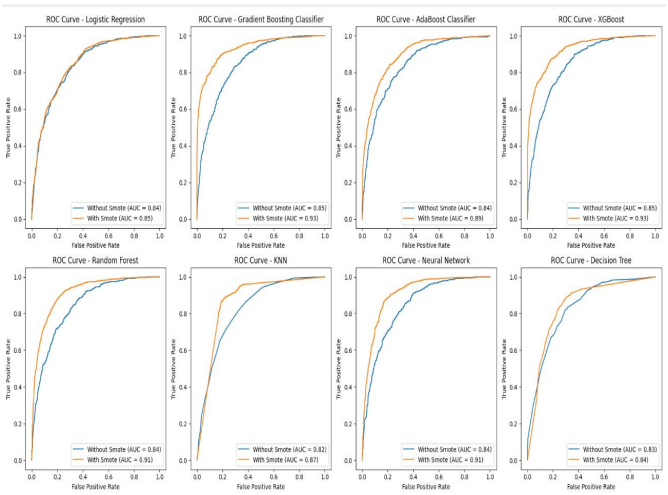


Figure below is the ROC curve for both the samples before using SMOTE and after using SMOTE :



V. CONCLUSION

On hyper-parameter tuned model :

Without using Smote: As data is imbalanced , best model is decided by AUC , and the best model without using smote is XG boost.
Using smote: We can select the model either by using f1 score, accuracy , AUC all are giving same model as best which is Gradient boosting classifier.

Here We can see that on both the samples with or without using SMOTE, boosting algorithm are giving the best results and the are giving the accuracy of 80% without SMOTE and 85% with SMOTE.

To achieve the higher accuracy we have done hyper-parameter tuning using Bayesian optimization which have improved the accuracy but for that we have to define optimally the range of hyper-parameters from which algorithm is going to select the best hyper-parameter.

V. REFERENCES

[1] B. Prabadevi, R. Shalini, B.R. Kavitha, Customer churnng analysis using machine learning algorithms.International Journal of Intelligent Networks 4 (2023) 145–154.

[2] Omar Adwan, Hossam Faris, Khalid Jaradat, Osama Harfoushi, Nazeeh Ghatasheh, Predicting customer churn in telecom industry using multilayer preceptron neural networks: modeling and analysis, Life Sci. J. 11 (3) (2014).

[3] S. Babu, N.R. Ananthanarayanan, V. Ramesh, A study on efficiency of decision tree and multi layer perceptron to predict the customer churn in telecommunication using WEKA, Int. J. Comput. Appl. 140 (4) (2016).

[4] Kamorudeen A. Amuda, Adesesan B. Adeyemo, Customers Churn Prediction in Financial Institution Using Artificial Neural Network, 2019, 11346 arXiv preprint arXiv:1912.

[5] Vrushabh Jinde, Savyanavar Amit, " customer churn prediction system using machine learning.", International Journal of Advanced Science and Technology 29 (5) (2020) 7957–7964.

[6] Irfan Ullah, Basit Raza, Ahmad Kamran Malik, Muhammad Imran, Saif Ul Islam, Sung Won Kim, A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector, IEEE Access 7 (2019) 60134–60149.

[7] Rahmat Yahaya, Opeyemi Aderiike Abisoye, Sulaimon Adebayo Bashir, An enhanced bank customers churn prediction model using A hybrid genetic algorithm and K-means filter and artificial neural network, in: 2020 IEEE 2nd International Conference on Cyberspac (CYBER NIGERIA), IEEE, 2021.

[8] Thanasis Vafeiadis, Konstantinos I. Diamantaras, Sarigiannidis George, K. Ch Chatzisavvas, A comparison of machine learning techniques for customer churn prediction, Simulat. Model. Pract. Theor. 55 (2015).