CHURN\_MODELLING Raunak Patil

UIN: 673045203

1. INTRODUCTION:

Customer retention is a top priority for many companies. Acquiring more customers can be more expensive than retaining customers. A churn model can be a tool that brings these elements together and provides insights and outputs that drive decision making across the organization.

Churn rate is calculated by dividing the number of customer cancellations within a period, by the number of active customers at the start of the same period. This can be used to predict new products behavior within a short amount of time or target a particular audience. The output of a churn model is to predict for the given circumstances, if the customer will not cancel or cancel the services provided by the organization.

In Machine Learning, this is a classic example of ‘Classification problem’. We can either use machine learning algorithms or use neural networks.

2. DATASET:

The dataset provided contains basic customer data of 10000 customers. There are 2 categorical features in Gender and Geography while the rest are integer based features.

The target variable is indicated in the column ‘Exited’ with values 1 being yes and 0 being not exited. It’s a fairly simple and clean dataset with no missing values or differently formatted cells.

3. DATA EXPLORATION:

The target value i.e. the ‘Exited’ column is highly imbalanced. The column has 2037 instances of ‘1’s and 7963 instances of ‘0’. Hence, there would be a considerable amount of bias introduced in the modeling stage. Hence, the first part is ran on the imbalanced dataset and the results achieved are typically not acceptable. The dataset is then under sampled and modeled which gives us acceptable accuracies.

The ‘Balance’ column is also imbalanced. It has around 4500 distinct values and around 5500 ‘0’s. This is a feature that might not be adding any value to the model.

4. RESULTS:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLASSIFIER** | **PRECISION [0]** | **PRECISION[1]** | **RECALL[0]** | **RECALL[1]** | **ACCURACY** | **F-SCORE[0]** | **F-Score[1]** | **ROC\_AUC** |
| SVM-Linear | 79.25 | 0.0 | 100.0 | 0.0 | 79.25 | 88.42 | 0.0 | - |
| SVM-RBF | 87.09 | 73.92 | 95.77 | 45.78 | 85.40 | 91.22 | 56.54 | - |
| SVM-RBF (1) | 87.09 | 73.92 | 95.77 | 45.78 | 85.40 | 91.22 | 56.54 | - |
| Decision\_Tree | 86.73 | 46.75 | 94.98 | 50.36 | 77.80 | 85.85 | 48.49 | 67.67 |
| Random\_Forest | 87.70 | 79.68 | 96.78 | 48.19 | 86.70 | 92.02 | 60.06 | 86.66 |
| XG\_Boost | 87.40 | 78.86 | 96.71 | 46.74 | 86.35 | 91.82 | 58.69 | 87.85 |
| NN\_model\_1 | 89.00 | 69.00 | 94.00 | 56.00 | 85.95 | 91.00 | 62.00 | - |
| NN\_model\_2 | 89.00 | 65.00 | 92.00 | 55.00 | 87.22 | 91.00 | 59.00 | - |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLASSIFIER** | **PRECISION [0]** | **PRECISION[1]** | **RECALL[0]** | **RECALL[1]** | **ACCURACY** | **F-SCORE[0]** | **F-Score[1]** | **ROC\_AUC** |
| SVM-Linear | 71.99 | 71.07 | 71.28 | 71.78 | 71.53 | 71.63 | 71.42 | - |
| SVM-RBF | 76.52 | 78.14 | 79.31 | 75.24 | 77.30 | 77.89 | 76.67 | - |
| SVM-RBF (1) | 76.52 | 78.14 | 79.31 | 75.24 | 77.30 | 77.89 | 76.67 | - |
| Decision\_Tree | 70.00 | 69.38 | 69.82 | 69.55 | 69.69 | 69.91 | 69.46 | 69.69 |
| Random\_Forest | 75.99 | 77.97 | 79.31 | 74.50 | 76.93 | 77.61 | 76.20 | 85.62 |
| XG\_Boost | 76.45 | 81.02 | 82.96 | 74.00 | 78.52 | 79.57 | 77.36 | 85.79 |
| NN\_model\_1 | - | - | - | - | - | - | - | - |
| NN\_model\_2 | 72.00 | 81.00 | 85.00 | 66.00 | 79.66 | 78.00 | 73.00 | 86.01 |

5. MODELLING:

There are 4 different types of models.

1. SVM:

The support vector machines are widely used classifier for classification and natural language processing. SVM's creates a hyper plane/s to classify data with accuracy controlled by the cost function, alpha, gamma and other parameters. There are 3 types of solvers used with RBF giving the best f1 score. This is probably because the data

is polarized.\*\* I tried to visualize the data on another book to see if the classification makes sense. I couldn't get it to work though. This would have enabled me to use appropriate SVM classifier and make distinct classification. The 'Radial Bias Function' AKA RBF solver is used and I tried to play with the cost functions to get better F-1 scores and accuracy. However, since it takes long to implement, I haven’t uploaded that in the code. The 'poly' function is also used since the data is binary to see if we could get better F1.

For the balanced data however, the precision and recall are acceptable. We are really interested in the precision [1] and recall [1] for which amongst all models, RBF and Poly are giving the highest recall[1] at 75.24%. The F-1 Score [1] is only second to XGB which is expected since SVM’s can only classify using linear methods.

1. DECISION TREE:

Decision Trees can be used as classifier or regression models. A tree structure is constructed that breaks the dataset down into smaller subsets eventually resulting in a prediction. There are decision nodes that partition the data and leaf nodes that give the prediction that can be followed by traversing simple IF..AND..AND….THEN logic down the nodes.

The root node (the first decision node) partitions the data based on the most influential feature partitioning. There are 2 measures for this, Gini Impurity and Entropy. The root node (the first decision node) partitions the data using the feature that provides the most information gain. Information gain tells us how important a given attribute of the feature vectors is. It is calculated as:

Information Gain=entropy (parent)–[average entropy (children)] Information Gain=entropy(parent)–[average entropy(children)]

Gini impurity is computationally faster as it doesn’t require calculating logarithmic functions, though in reality which of the two methods is used rarely makes too much of a difference.

Based on the above explanations, the results for the imbalanced dataset gives us decent values for the [0]’s but unacceptable scores for [1]’s. The precision [0] score is 86.73% while as precision [1] is 46.75%. Similarly, the recall [0] is 94.98% while the recall [1] is 50.35%. This is primarily because the information gain is not adequate to drive similar looking points which are misclassified. We could try changing the entropy of the target class by using appropriate transformations to the dataset to see if we could increase the values.

For the balanced dataset, the F-score is 69.50% which is decent but not the best. The AUC is 69.69%. The decision tree classifier may be used where there are more distinct features.

1. Random Forest:

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It is a bagging technique and involves bootstrapping as well. It then aggregates the votes from different decision trees to decide the final class of the test objects to get a more accurate and stable prediction. It searches for best feature among a random subset of features. This results in wide diversity that generally results in a better model. However, increasing number of estimators could over fit but it is generally accurate. It will be computationally expensive.

In our case, on the imbalanced dataset, we get acceptable precision [1] but unacceptable recall [1] with the values being 79.68% and 49.16%. In the balanced dataset we can see a huge improvement, with precision [1] is 77.97% and recall [1] is 75.14%. This is also due to the precision-recall curve. They are inversely related and if precision goes up, recall goes down and vice-versa. They are fairly acceptable though.

1. XG\_BOOST: (BOOSTING TECHNIQUE)

The gradient boosting method is widely used. The term ‘Boosting’ refers to a family of algorithms which converts weak learner to strong learners. To convert weak learner to strong learner, we’ll combine the prediction of each weak learner using methods like:   Using average/ weighted average or considering prediction has higher vote. To find weak rule, we apply base learning (ML) algorithms with a different distribution. Each time base learning algorithm is applied, it generates a new weak prediction rule. This is an iterative process. After much iteration, the boosting algorithm combines these weak rules into a single strong prediction rule. For choosing the right distribution, here are the following steps:

*Step 1:*  The base learner takes all the distributions and assigns equal weight or attention to each observation.

*Step 2:* If there is any prediction error caused by first base learning algorithm, then we pay higher attention to observations having prediction error. Then, we apply the next base learning algorithm.

*Step 3:* Iterate Step 2 till the limit of base learning algorithm is reached or higher accuracy is achieved.

Finally, it combines the outputs from weak learner and creates a strong learner which eventually improves the prediction power of the model. Boosting pays higher focus on examples which are mis-classiﬁed or have higher errors by preceding weak rules. There are many boosting algorithms which use other types of engine such as:

1. AdaBoost (Adaptive Boosting)
2. Gradient Tree Boosting
3. XGBoost

I have used XGBOOST as it is the fastest of all 3. The pre-defined base model is used to classify. I tried changing in the trial notebook some parameters but all of them were giving the same accuracy. I was led to believe, there may be something wrong with the dataset I am feeding to the model. But we can see it is not giving enough accuracy. The precision [1] is 78.86 and recall [1] is 46.74% for the unbalanced dataset. This brings the F-score down to 58.69% with roc being 87.85%. We should take a note that, even if the roc is high enough, the recall is very low which is of interest in this model.

However, for the balanced dataset we see the F-scores have greatly improved and is the best I have achieved among all models. The F-score [1] is 77.36 with roc being 85.68 which is a very small decrease. These are acceptable.

1. Neural NETWORKS:

A supervised neural network can be thought of as a human brain where, we take input, learn (process, update) about the input and produce output usually resulting in some form of human activity. There are some steps involved in modeling a neural network and they are:

1. Model Initialization: Random initialization of a model is common practice. The rationale behind is that from wherever we start, if we are perseverant enough and through an iterative learning process, we can reach the pseudo-ideal model.
2. Forward Propagation: There are 3 types: Forward, backward and both. This is related to updating of the weights at each node/layer of the model going forward.
3. Loss Function: It is a performance metric on how well the nn model manages to reach its goal of generating outputs as close as possible to desired values.
4. Differentiation: This is how the cost function moves toward the local or global optimum. It could be gradient descent or genetic algorithm.
5. Back-Propagation: After the cost function has reached a new point, the output function is back propogated to check for errors.
6. Weight-update: After one iteration, all weights are updated for all nodes.
7. Iterate until convergence: The process is repeated until the function reaches global or local minimum depending on the stopping criteria.

There are many hyper parameters that can be defined. In this model, I have used

1. ‘kernel\_init’ to have uniform distribution
2. Activation function: Relu
3. Regularizer : L2
4. Loss: Binary cross entropy
5. Optimizer: Adam

The results are also the best among all models. However, the veracity of results for the unbalanced dataset cannot be trusted even though they have acceptable F scores such as 91% and 62%.

The difference between the accuracies of model 2 is significant. For the unbalanced dataset, since I have used a lot of neurons, the model may be prone to over fit. It has F scores of 91 and 59 as compared to 78 and 73 for the structured dataset. There difference is really notable in the recall scores. For the unbalanced dataset the recall [1] scores are 92% and 55% while for the balanced dataset it is 85% and 66%.

The ROC is also the best among all models, which comes to about 86.01%.