Playing With Data

Krantik Das

May 2023

1 Overview

In this project, it is objected to build a classification model of the data set. It has 216 independent features, from X0 to X215. The UID is the Unique identifier for each row and, Target_ChurnFlag is the binary target variable. The steps that have been followed by me for this are:

Data Preparation:

1. Load the dataset into Python environment.
2. Performing necessary data cleaning, preprocessing, and feature engineering.
3. Handling missing values, encode categorical variables, normalize or scale numerical features, etc.

Train-Test Split:

1. Splitting the dataset into training and testing sets.
2. The training set will be used to train the model.
3. The testing set will be used to evaluate the model's performance.

Model Selection:

1. Choosing an appropriate classification algorithm for the problem.
2. Some common algorithms include Logistic Regression, Decision Trees, Random Forests, Gradient Boosting, Support Vector Machines, or Neural Networks. In this case, Random Forests have been used.

Model Training:

1. Training the selected classification model using the training data.
2. Fitting the model to the training set.
3. Optimizing the model's parameters to minimize the classification error.

Model Evaluation:

1. Evaluating the trained model using the testing set.
2. Calculating performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to assess the model's predictive ability.

Model Optimization:

1. Fine-tuning the model by adjusting hyperparameters or using techniques like cross-validation to improve its performance.
2. Prevent overfitting and optimize the model's generalization ability.

Prediction:

1. Using the trained model to make predictions on new, unseen data.

2 Data Preparation

The given dataset contains:

Data Type	Count
int64	97
float64	89
object	30
bool	1

The various data types need to be handled accordingly. Since we want to fit into a model, we need to convert the non numeric rows into numeric. For this, the rows either can be dropped entirely from the dataset, or if the variable is a categorical variable, it can be converted into a dummy variable.

We removed the target variable from the dataset which is, **Target_ChurnFlag**. Further, as UID or Unique Identifier for wach row is just an identification feature and not a explanatory variable, we have excluded that as well. Further we dropped a series of variables from the dataset, but none of them were in line with the analysis of the data. Next, we fill in the null values with 0 or any other imputation techniques, such as mean. Next, converting categorical variables into numerical variables.

3 Train-Test Split

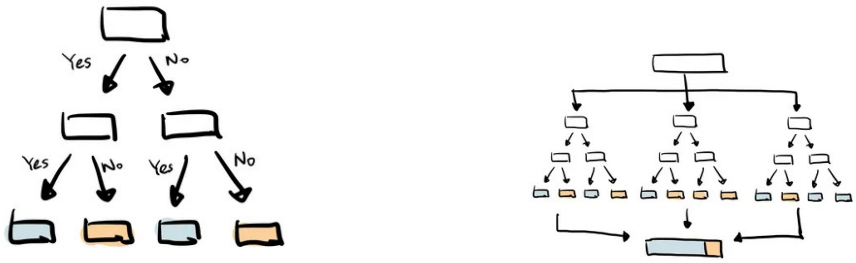
Moving on with our analysis, we have divided our dataset into two parts in a 80:20 ratio. 80% of the dataset has been used as the training set on which the model has been trained on, and the remaining 20% measures it's performance.

Classification algorithm falls under the category of supervised learning, so dataset needs to be split into a subset for training and a subset for testing (sometime also a validation set). The model is trained on the training set and then examined using the testing set.

4 Model Selection

For the analysis, Random Forests has been chosen. A random forest is a type of ensemble method that consists of a collection of decision trees. It is called a "forest" because it combines the predictions of multiple trees to make a final prediction. Random forests use a technique called bagging, where each tree is trained on a random subset of the original dataset. The final prediction is determined by taking the majority vote from the individual tree predictions.

Random forests have several advantages over individual decision trees. They tend to have better generalization performance, meaning they can make more accurate predictions on unseen data. However, the tradeoff is that random forests are less interpretable compared to a single decision tree, as they involve multiple layers of decision-making.



(a) A Single Decision Tree.

(b) A Random Forest.

Figure 1: A Sigle Decision Tree vs Random Forest

In summary, a random forest is an ensemble method that leverages the power of decision trees by aggregating their predictions. It improves generalization but sacrifices interpretability due to its complexity.

5 EDA

Histogram, grouped bar chart and box plot are suitable EDA techniques for classification machine learning algorithms.Histogram is used for all features, because all features have been encoded into numeric values in the dataset.

6 Model Training

This is one of the most important steps of the analysis. We train our model on the dataset. So, during this process, the model that we have chosen is Random Forests and we have used the RandomForestClassifier() command to do so. The objective of this exercise is to predict Target_ChurnFlag. Therefore, the first task is to separate the input features (independent variables — X) and the label (dependent variable — y). Secondly, both features and labels are broken down into a subset for training and another for testing. As the result, four portions are returned, X_train, X_test, y_train, and y_test. To achieve this, we introduce the train_test_split function and specify the parameter test_size.

Thanks to scikit-learn, we can skip the laborious task of implementing all the mathematical algorithms ourselves. Instead, we can import the LogisticRegression class from the sklearn library and train the model with our training data. However, we still have the freedom to customize the model by specifying various parameters. For more complex machine learning models, we often need to tune the hyperparameters by searching through different values to find the best combinations.

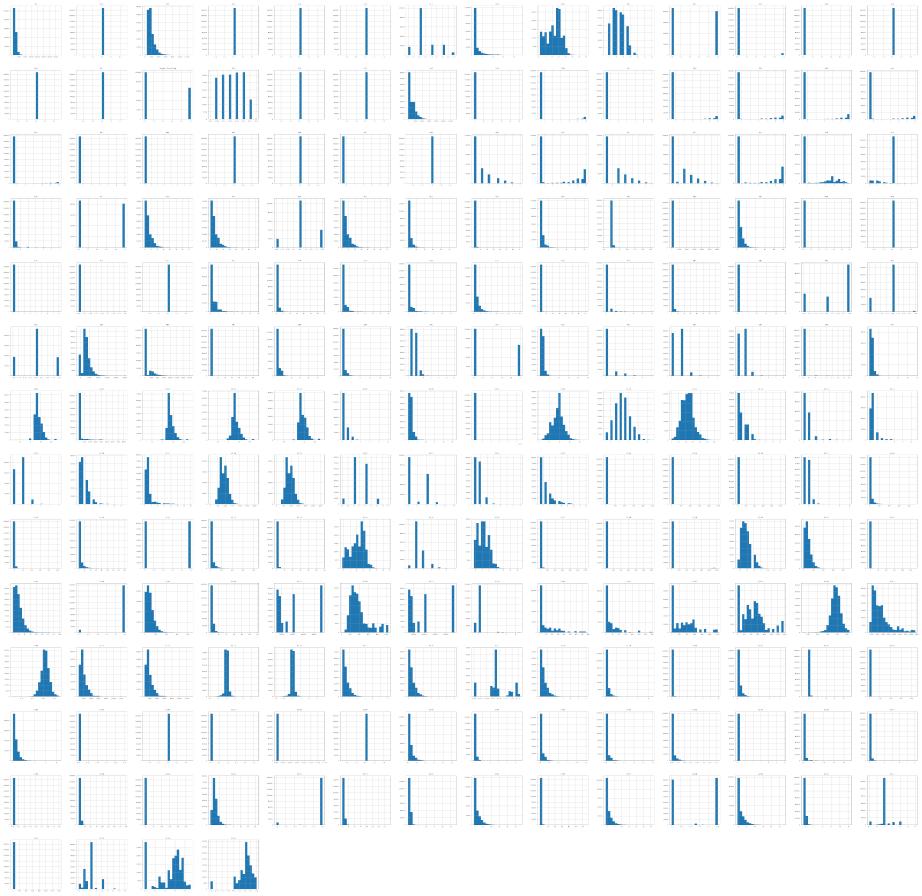


Figure 2: Histogram for EDA

7 Model Evaluation

ROC, AUC, Confusion Matrix, and Accuracy are widely used for evaluating the model. These metrics are based on calculating the differences between the predicted y-values (y_pred) by the model and the actual y-values (y_test) of the test set. There are four possible scenarios when comparing these differences:

True Positive: It refers to the cases where the model predicts positive (e.g., raining) and the actual observation is also positive (raining).

True Negative: It refers to the cases where the model predicts negative (e.g., not raining) and the actual observation is also negative (not raining).

False Positive: It refers to the cases where the model predicts positive (raining) but the actual observation is negative (not raining).

False Negative: It refers to the cases where the model predicts negative (not raining) but the actual observation is positive (raining).

7.1 Confusion Matrix

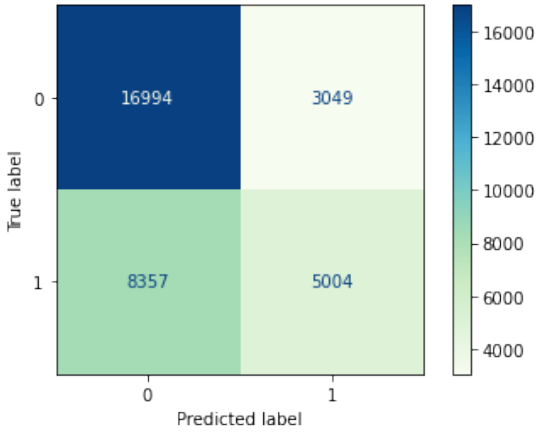


Figure 3: Confusion Matrix

I used `plot_confusion_matrix()` to provide a visual representation that clearly indicates the counts of the four scenarios mentioned above. As shown, the true negative is 16994 cases, suggesting that the model is good at predicting the target variable as 0, when it is actually 0. However, it still needs improvement on the true positive rate, hence successfully predicting the out of sample data (only 5004 cases).

7.2 Accuracy

Accuracy is the most straightforward indicator of model performance. It measures the percentage of accurate predictions, which is calculated by dividing the sum of true positives and true negatives by the total number of predictions:

$$accuracy = \frac{truepositive + truenegative}{truepositive + falsepositive + falsenegative + falsepositive}$$

7.3 ROC AUC

ROC (Receiver Operating Characteristic) is a plot of the true positive rate against the false positive rate at various classification thresholds. It provides a graphical representation of the trade-off between sensitivity (true positive rate) and specificity (1 - false positive rate) for a binary classification model.

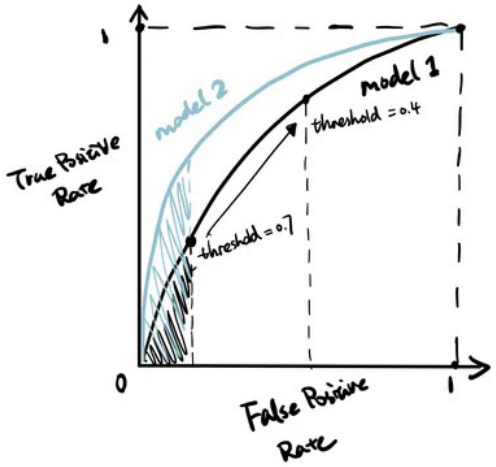


Figure 4: ROC AUC Curve

AUC (Area Under the Curve) is the area under the ROC curve. It quantifies the overall performance of the model, with higher AUC values indicating better discrimination ability. AUC values range from 0 to 1, where 0.5 represents a random classifier and 1 represents a perfect classifier.

In this model, accuracy is 0.658992934977847, and the ROC AUC score is 0.6124350030116187. These values indicate the performance of the model in terms of accuracy and the area under the receiver operating characteristic curve (ROC AUC). The accuracy represents the percentage of correctly predicted samples, while the ROC AUC score measures the model's ability to distinguish between positive and negative samples.

An accuracy of 0.6589 and an ROC AUC score of 0.6124 suggest that the model's performance is modest. It means that the model is making correct

predictions for approximately 65.9% of the samples, and it has a moderate ability to distinguish between positive and negative samples.

8 Prediction

Finally, the model has been used to predict the values of the target variable, Target_ChurnFlag. An array of 186 zeroes has been created to act as an input variable. By optimizing accordingly, we can see the various predictions.

Class	**Precision**	**Recall**	**F1-score**	**Support**
0	0.67	0.85	0.75	20043
1	0.62	0.37	0.47	13361
Accuracy			0.66	
Macro Avg	0.64	0.61	0.61	33404
Weighted Avg	0.65	0.66	0.64	33404

Precision: Precision measures the accuracy of the positive predictions. In this case, the precision for class 0 is 0.67, which means that out of all the instances predicted as class 0, 67% of them are actually class 0. For class 1, the precision is 0.62, indicating that 62% of the instances predicted as class 1 are indeed class 1.

Recall: Recall, also known as sensitivity or true positive rate, measures the ability of the model to correctly identify the positive instances. The recall for class 0 is 0.85, indicating that the model correctly identified 85% of the actual class 0 instances. For class 1, the recall is 0.37, meaning that only 37% of the actual class 1 instances were correctly identified.

F1-score: The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, taking both precision and recall into account. The F1-score for class 0 is 0.75, and for class 1, it is 0.47.

Support: Support represents the number of instances of each class in the test dataset. In this case, there are 20,043 instances of class 0 and 13,361 instances of class 1.