Introduction

Using the data provided for customer churning at IBM the goal of this report is to identify insights and tactics to aid in creating a customer retention program. The report will be broken up into two key tasks, the first involving the creation of a white box model (Interpretable), and the second a (Blackbox). The white box model aims to provide a clear set of rules that highlight why people have churned, while the black box aims for greater model prediction performance of future churners/No churners.

Understanding the data set

The data set consists of 20 predictive attributes and a target. The 20 predictive attributes are a mix of numerical data points and binary/categorical points, 16 categorical and 4 numerical. The target column set says if a particular customer has left in the last month or not.

Interesting Attribute details

- Of those that have left/churned 72% are Senior citizens.
- The average tenure of those who have been retained is 37 while the average of those who left is 18.
- 86% of those that churned had month to month contracts.
- The target attribute ‘churn’ is skewed to ‘No’ over ‘Yes’ by 73% to 27%. Our models will be much better at predicting ‘No_Churn’ over ‘Yes_Churn’
- 83% of churners had no dependents while ‘no churners’ was 65% no dependents.

Task-1: Interpretable Model

a) Preparing the data for modelling

In the building of the model, I’m going to use the full list of attributes to build a preliminary model. Once this model has been assessed/scored via a confusion matrix I will begin the ‘Pruning the tree’ by tuning the hyperparameter alpha (complexity penalty).

Another alternative that could be explored is forward feature selection which works in the opposite way to pruning. You begin with the attribute with lowest gini index, set that as the root, loop through all combinations of these two, select the lowest gini combo, and repeat.

Furthermore, the categorical data needs to be converted into numerical values. I used one-hot method to transform the categorical data.

b) Methodology

The methodology for this Decision Tree build is to begin with a ‘blank canvas’ of default hyperparameters as well as the full set of attributes. Once this has been built and assessed I will prune the tree by tuning the hyper parameter alpha. This alpha will then become the parameter for the final model.

From the pruned tree we should be able to see the key rules that offer the highest purity and offer insights that will help in the creation of a customer retention program.

c) Building and evaluating models

First, we split the data into train and test samples, we then train a decision tree with default parameters on the full set of attributes in the test set. We then evaluate the preliminary model using a confusion matrix.

At this point the model is quite overfitted. We can tune this down using the hyperparameter alpha.

After completing the alpha tuning, we assess the precision and recall of the final model.

d) Conclusion

Precision improved from 60% to 74% between the pruned and un-pruned models, and true negative (Churn_Yes) improved from 62% to 65% as well.

Good rules for the classifications are:

Churn_No

- (SeniorCitizen=0)->(tenure<6.5)->(onlineSecurity_No=0) ; Purity=78%
- (SeniorCitizen=0)->(tenure<6.5)->(Contract_Month-toMonth=0); Purity=94.7%

Churn_Yes

- (SeniorCitizen=1)->(Contract_Month-toMonth=1)->(tenure<=16.5yrs); Purity=77.4%
- (SeniorCitizen=0)->(tenure<6.5)->(onlineSecurity_No=1)->(InternetService_Fiber optic=1)->(TotalCharges<=89.1); Purity=0.663

Task-2

a) Preparing the data for modelling

We will build on from the model built in the previous steps and use all the attributes to begin with. We will be using AdaBoost and Random Forest to build our ensemble models.

b) Methodology

For AdaBoost we will use the best model from the first task as our weak learner and aim to improve upon its performance.

Furthermore, we will create a fresh Random Forest Classifier and use GridSearchCV to refine the parameters.

c) Building and evaluating models

Keeping the same data splits as before, we train a decision tree using the alpha parameter found in task one and then apply the adaboost function.

For the random forest we create a parameter grid of the key parameters (n_estimators, max_features ect), run the GridSearchCV function to find the optimal parameter values, and then fit the random forest to the training data.

For here we assess the precision and recall values for both models.

d) Conclusion

The Random Forest model had the highest Precision at around 78% vs AdaBoost at 63%, whereas Recall was best for AdaBoost at 61% vs 51%. Overall, each individual model didn't quite

Overall Conclusion

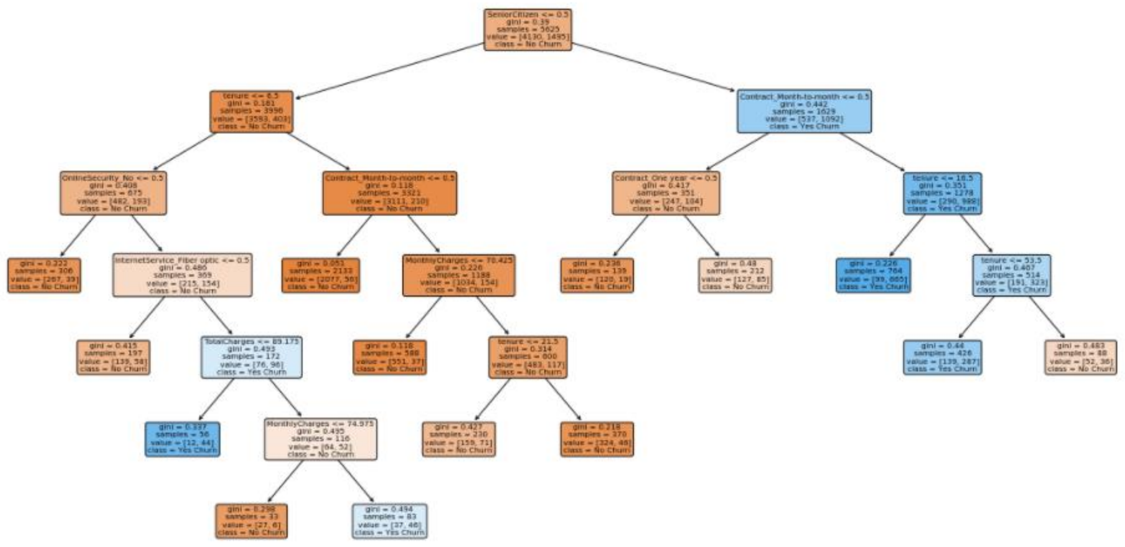
Overall Adaboost reduced the precision of the model, so I will retain the original decision tree here. Since a purity of 75%+ is considered a good rule, for people who have 'churned' the key rule would be that they are senior citizens, are on a month-to-month contract, and a tenure of <=16.5 years. We could recommend a 'seniors' program' and try and move them from month-to-month contracts to something longer but still flexible.

The Random Forest was the best performer with precision at 78% but due to being a 'blackbox' we can't visualise and key rules.

Appendix

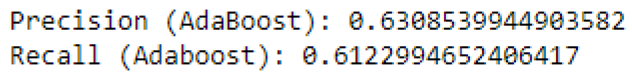
Task-1

Pruned Decision Tree

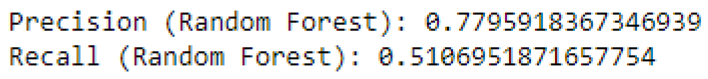

SeniorCitizen <= 0.5 gini = 0.39 samples = 5625 value = [4130, 1495] class = No Churn
tenure <= 6.5 gini = 0.181 samples = 3996 value = [3593, 403] class = No Churn
Contract_Month-to-month <= 0.5 gini = 0.442 samples = 1629 value = [537, 1092] class = Yes Churn
OnlineSecurity_No <= 0.5 gini = 0.408 samples = 675 value = [482, 193] class = No Churn
Contract_Month-to-month <= 0.5 gini = 0.118 samples = 3321 value = [3111, 210] class = No Churn
Contract_One year <= 0.5 gini = 0.417 samples = 351 value = [247, 104] class = No Churn
tenure <= 16.5 gini = 0.351 samples = 1278 value = [290, 988] class = Yes Churn
gini = 0.222 samples = 306 value = [267, 39] class = No Churn
InternetService_Fiber optic <= 0.5 gini = 0.486 samples = 369 value = [215, 154] class = No Churn
gini = 0.051 samples = 2133 value = [2077, 56] class = No Churn
MonthlyCharges <= 70.425 gini = 0.226 samples = 1188 value = [1034, 154] class = No Churn
gini = 0.236 samples = 139 value = [120, 19] class = No Churn
gini = 0.48 samples = 212 value = [127, 85] class = No Churn
gini = 0.226 samples = 764 value = [99, 665] class = Yes Churn
tenure <= 53.5 gini = 0.467 samples = 514 value = [191, 323] class = Yes Churn
gini = 0.415 samples = 197 value = [139, 58] class = No Churn
TotalCharges <= 89.175 gini = 0.493 samples = 172 value = [76, 96] class = Yes Churn
gini = 0.118 samples = 588 value = [551, 37] class = No Churn
tenure <= 21.5 gini = 0.314 samples = 600 value = [483, 117] class = No Churn
gini = 0.44 samples = 426 value = [139, 287] class = Yes Churn
gini = 0.483 samples = 88 value = [52, 36] class = No Churn
gini = 0.337 samples = 56 value = [12, 44] class = Yes Churn
MonthlyCharges <= 74.975 gini = 0.495 samples = 116 value = [64, 52] class = No Churn
gini = 0.427 samples = 230 value = [159, 71] class = No Churn
gini = 0.218 samples = 370 value = [324, 46] class = No Churn
gini = 0.298 samples = 33 value = [27, 6] class = No Churn
gini = 0.494 samples = 83 value = [37, 46] class = Yes Churn

Task-2

Output



```
Precision (AdaBoost): 0.6308539944903582
Recall (Adaboost): 0.6122994652406417
```



```
Precision (Random Forest): 0.7795918367346939
Recall (Random Forest): 0.5106951871657754
```

Reference List

J. Starmer, Classification Trees in Python from Start to Finish, StatQuest, 07/06/21, accessed on 30/07/21