Use Case: Churn Prediction

White Papers

OBJECTIVE

The objective of this project was to build machine learning models that use customer behavior metrics to predict churn for a multinational telecommunications company. Identifying customers who are likely to churn is a valuable task that would allow the company in question to deliver custom and focused promotion programs, segment customers based on behavior metrics, minimize acquisition costs, maximize marketing efficiency, and improve overall retention.

DATA

To carry out our objective, we extracted a data frame with 3,333 observations and 20 potentially relevant variables. Each observation corresponded to an individual customer who either churned or remained with the company. The target variable (label) of the dataset was imbalanced, as 14.5% of the observations churned and 85.5% of the observations did not churn.

PREPROCESSING

Preprocessing steps included handling missing values, encoding categorical features, and encoding the outcome variable (label). The reason that we converted the categorical string features into encoded numerical features was that machine learning models tend to perform better when text classes are translated into binary inputs.

In addition, we performed a holistic exploratory data analysis to understand the characteristics and nature of the data.

MODEL BUILDING AND EVALUATION #1: BENCHMARK

Although the dataset was imbalanced, as it had a clear majority of observations that did not churn, we first created a set of benchmark models in which we did not equalize the label classes by oversampling or undersampling the data. In addition, for these benchmark models, we did not perform feature selection techniques and, therefore, used all 19 features to predict churn.

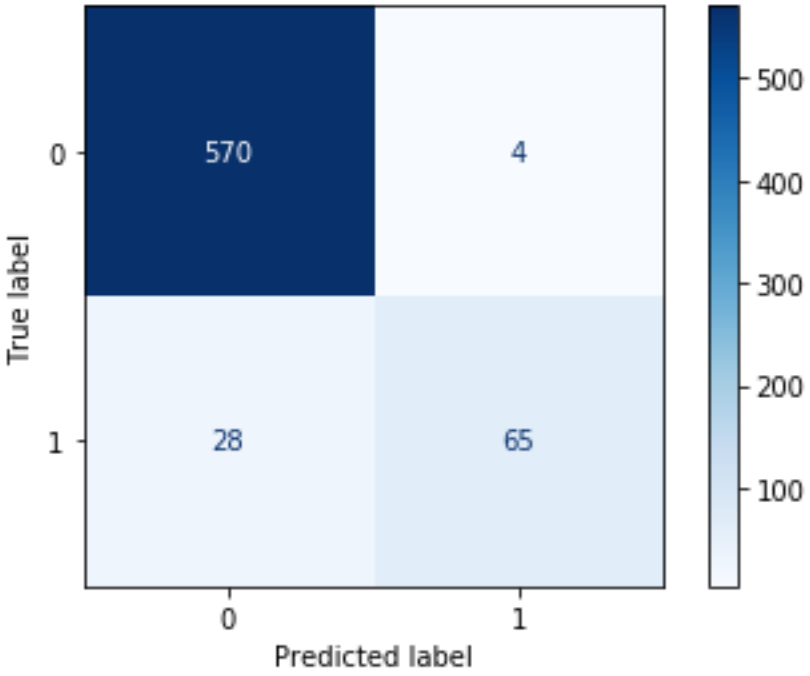
Next, we randomly split the data frame into a training set (80%) and a testing set (20%). Subsequently, we built the following machine learning algorithms on the training set: (1) Logistic Regression, (2) Support Vector Machines, (3) Multi-Layer Perceptron (Artificial Neural Networks), (4) Random Forests, and (5) Gradient Boosting.

For each of the five algorithms, we tested hundreds of iterations by tuning the model hyperparameters. To select the best model within each algorithm, we performed cross-validation with k=5 and selected the model with the highest cross-validation classification accuracy. Therefore, at the end of the model building stage, we chose a total of five models from a set of over thousand models. More specifically, we chose the best Logistic Regression model, the best Support Vector Machines model, the best Multi-Layer Perceptron model, the best Random Forests model, and the best Gradient Boosting model.

Next, we used each of these models to predict the observations in the testing set. Note that this step emulates predicting real data, as these models have never seen these observations before. After predicting each observation in the testing set, we compared the predictive accuracies of the five models. The metrics that we used were overall classification accuracy, precision, and recall.

The best overall model was the Gradient Boosting model with a learning rate of 0.1, max depth of 7, and number of estimators of 500 (amongst multiple other hyperparameters). The results were as follows:

- Accuracy: 95.2%
- Precision: 94.2%
- Recall: 69.9%

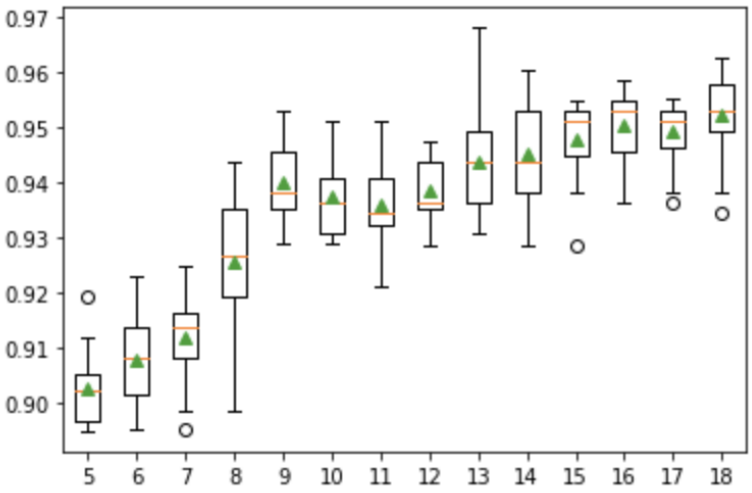


MODEL BUILDING AND EVALUATION #2: FEATURE SELECTION

In the second model building phase, we added an extra layer of complexity relative to the benchmark models. To be specific, we performed feature selection to isolate the most important features in terms of predictive power. In this phase, we did not address the imbalanced nature of the data frame.

To conduct feature selection, we used a wide variety of techniques on the training data. It is crucial to use feature selection techniques only on the training data and not the entire dataset to avoid overfitting the models to the unseen testing data. First, we looked at the variable importance from the Random Forest model previously trained. Second, we performed the following feature selection techniques: F-Classif, Chi-square, and F-Regression. All feature selection techniques were consistent, as they selected the same top features. Third, we used the ExtraTreesClassifier to perform feature selection. Fourth, we used the Recursive Feature Elimination (RFE) technique. As we found the cross-validation accuracy to be higher (with Random Forests) using the features given by the RFE technique, we proceeded with the RFE method.

The next step was to evaluate the optimal number of features to select using RFE. Accordingly, we iteratively added features based on their predictive power, starting from the top 1 feature until all the features were added. For each iteration, we logged the cross-validation classification accuracy. The figure below shows that using 18 variables yielded the highest cross-validation accuracy.

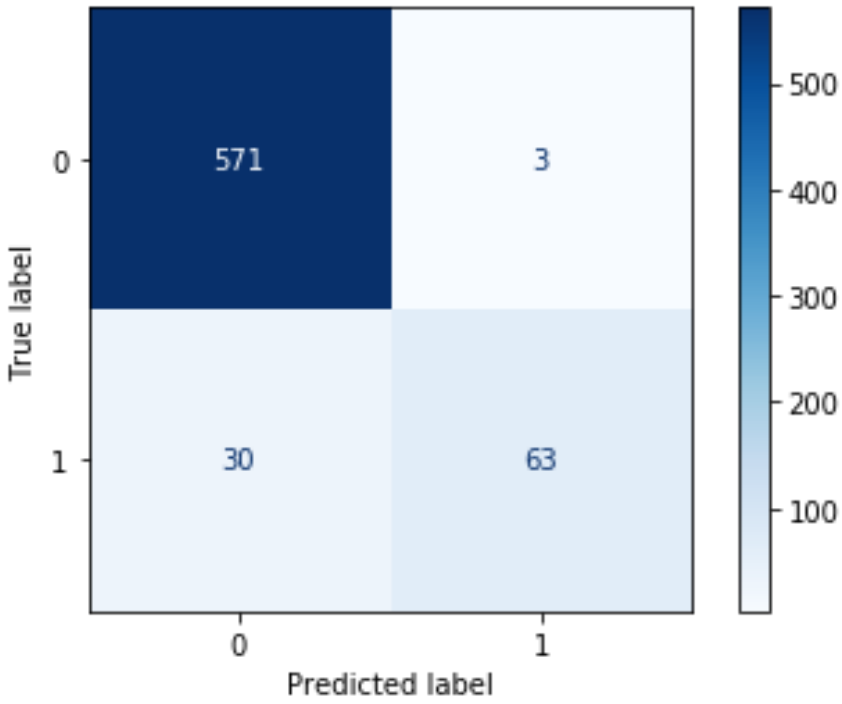


The variable dropped was consistent with the worst variable (in terms of predictive power) of the other feature selection techniques.

Next, we performed the same steps as the previous phase: we built hundreds of iterations of the five machine learning algorithms highlighted above (to tune hyperparameters), selected the best model in terms of cross-validation accuracy within each algorithm class, and evaluated the top five models on the testing set.

The best overall model was the Gradient Boosting model with a learning rate of 0.1, max depth of 5, and number of estimators of 250 (amongst multiple other hyperparameters). The results were as follows:

- Accuracy: 95.1%
- Precision: 95.5%
- Recall: 67.7%



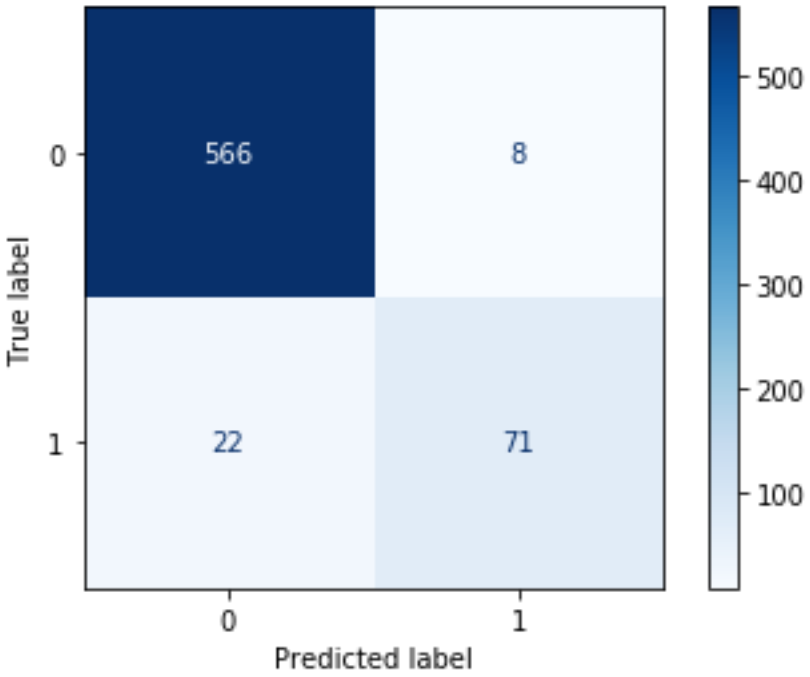
MODEL BUILDING AND EVALUATION #3: SMOTE

In the third model building phase, we addressed the imbalance problem of our dataset by using an oversampling technique called Synthetic Minority Over-Sampling Technique (SMOTE). This technique was essentially used to equalize the two label classes to 50% each. However, in this phase, we used all the variables and not the top 18 variables given by feature selection. This is because using all features gave us a higher predictive accuracy in the previous phases. It is essential to note that the oversampling was only done for the training set observations and not the testing set records, because we want our models to predict true and not synthetic observations.

As such, we built hundreds of iterations of the five machine learning algorithms (while tuning hyperparameters), selected the best model in terms of cross-validation accuracy within each algorithm class, and evaluated the top five models on the testing set.

The best overall model was the Gradient Boosting model with a learning rate of 0.1, max depth of 7, and number of estimators of 500 (amongst multiple other hyperparameters). The results were as follows:

- Accuracy: 95.5%
- Precision: 89.9%
- Recall: 76.3%



MODEL BUILDING AND EVALUATION #4: AZURE ML

In the fourth model building phase, we leveraged Microsoft Azure's Machine Learning Studio to recreate the previous machine learning models built. We experimented with various combinations of pipelines: (1) No feature selection and no SMOTE, (2) Feature Selection and no SMOTE, (3) SMOTE and no feature selection, and (4) Feature Selection and SMOTE. For each of these pipelines, we built hundreds of models based on the five machine learning algorithms outlined above with the objective of tuning hyperparameters to maximize predictive accuracy.

Next, we used each model to predict the observations in the testing set and evaluated their performance. Across the four pipelines, the best model was a Gradient Boosting model which was built on a training set without feature selection and without SMOTE. This model had the following accuracy measures:

- Accuracy: 95.4%
- Precision: 96.1%
- Recall: 72.3%

MODEL BUILDING AND EVALUATION #5: CUSTOM AZURE ML MODEL

In the fifth and final model building phase, we created a custom model on Azure's Machine Learning Studio. This custom script consisted of an untrained Gradient Boosting model with the optimal hyperparameters, no feature selection, and no oversampling technique.

Subsequently, we trained this custom model on the training set, predicted the unseen observations in the testing set, and evaluated the model's accuracy. This model produced the best results amongst all of the models created, which are as follows:

- Accuracy: 95.8%
- Precision: 85.95%
- Recall: 80.2%

This model accurately predicted whether a customer would churn or not with an accuracy of 95.8%. More specifically, it correctly identified 80.2% of all the customers that actually churned, thereby yielding the highest recall of all the models created.

MODEL DEPLOYMENT

Using the model built in the fifth phase, we deployed a web service inference pipeline that allowed the end-user to make real-time predictions and provided the option to make batch predictions.

Since this model predicted 666 unseen testing observations with an accuracy of 95.8%, we can expect it to perform with a similar accuracy for real, new customer data.