Telecommunications Customer Churn Report

edX user jojo21250

9/22/2020

Overview

The aim of this project is to predict customer churn at a fictional telecommunications company. Predicting customer churn serves two functions. First, it allows the company to assess its subscriber base and more accurately forecast future revenue. Second, it provides the company with valuable intelligence about which customers should be targeted for retention efforts.

This report explores a fictional dataset of customer information and evaluates several prediction methods. The data includes individualized information about past customers, such as a customer's unique ID, which services were purchased, and whether or not the customer churned. It contains 7043 observations (each customer is one row) of 21 variables.

An overview of the variables and their possible values:

- **personal information**
  - customerID (string of characters)
  - gender (male/female)
  - senior citizen (yes/no)
  - partner(yes/no)
  - dependents (yes/no)
- **service information**
  - tenure (numeric value from 0 - 75)
  - phone service (yes/no)
  - multiple lines (yes/no/no phone service)
  - internet service (DSL/fiber optic/no)
  - online security (yes/no/no internet service)
  - online backup (yes/no/no internet service)
  - device protection (yes/no/no internet service)
  - tech support (yes/no/no internet service)
  - streaming TV (yes/no/no internet service)
  - streaming movies (yes/no/no internet service)
- **billing information**:
  - contract type (month-to-month/one year/two year)
  - paperless billing (yes/no)
  - payment method (electronic check/mailed check/automatic bank transfer/automatic credit card payment)
  - monthly charges (numeric value 10 - 120)
  - total charges (numeric value up to 9000)
  - churn (yes/no)

The data is provided on the GitHub page of IBM employee Scott D'Angelo. You can find the relevant repository here. The contents of the repository fall under the Apache 2.0 license, which can be viewed here.

Methods and Analysis - Part 1: Data Preparation and Exploration

The first step is to load the data and perform some basic preparation steps. Many of the variables contain words as observation, such as "Yes" and "No". These should be treated as factors, not characters.

```
#Change class of character variables to factors
dat <- dat %>% mutate_if(is.character,as.factor)
```

The SeniorCitizen variable contains observations such as "1" and "0", which indicate whether or not a customer is a senior citizen. This is a numeric value; it should be changed to factor as well.

```
#Change class of SeniorCitizen to factor,
#because it is just a TRUE/FALSE style observation represented by 1/0,
#and NOT a representation of some quantity
dat <- dat %>% mutate(SeniorCitizen = as.factor(SeniorCitizen))
```

A quick check for NAs shows that there are eleven.

```
sum(is.na(dat))
```

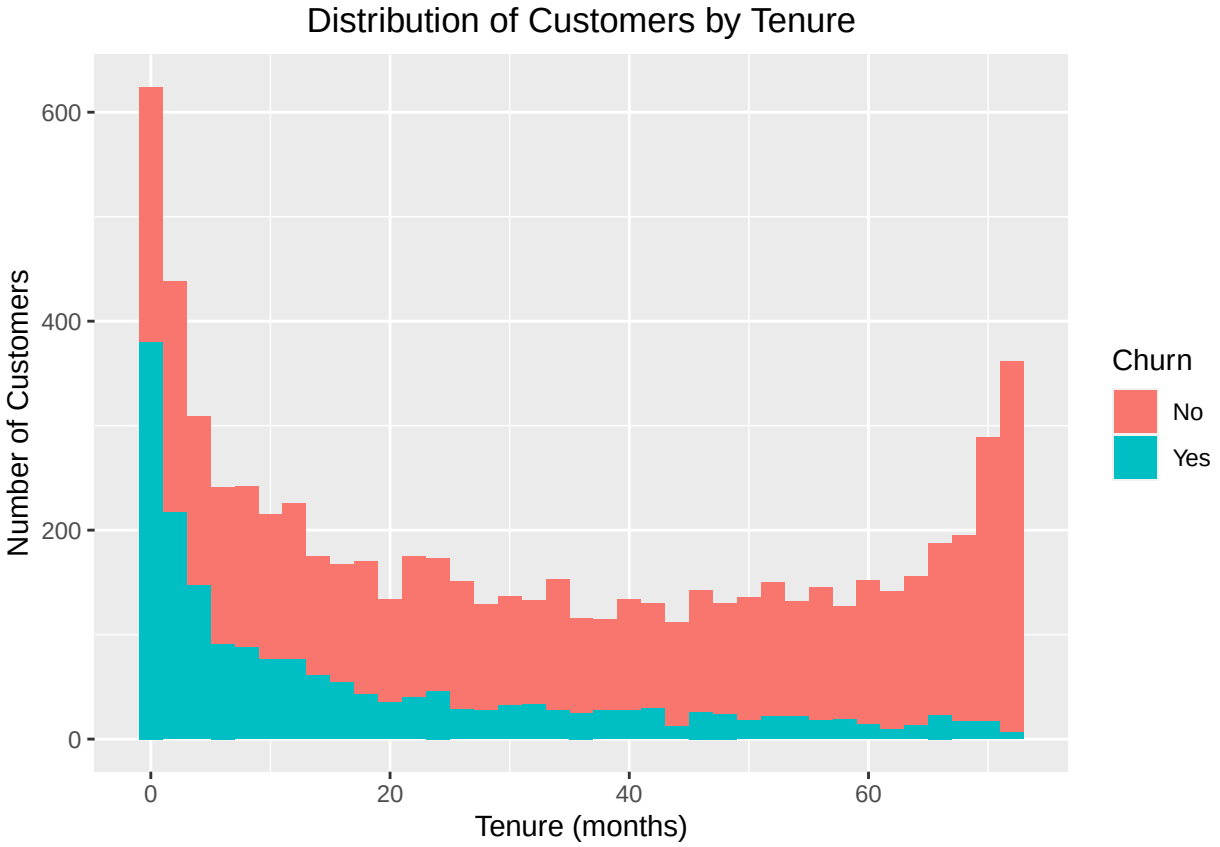
```
## [1] 11
```

Eleven NAs in 7043 observations is not very significant. However, machine learning algorithms require some way of dealing with NAs. Any method will have a minimal impact on the models and the predictions produced. For the sake of convenience, this report will use the **na.roughfix** method from the randomForest package. This method imputes the median value for missing numerics and the mode value for missing factors. It does not remove or ignore rows with NAs.

Below is a snippet of the data. Only the first nine columns are shown due to space limitations.

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL
3668-QPYBK	Male	0	No	No	2	Yes	No	DSL
7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic
9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic

Next, the data is explored with some visualizations. The chart below shows the distribution of customers by tenure (the length of time they have been a customer). A significant portion of the customer base is relatively new. Without even looking at exact numbers, it is apparent that much of the churn occurs among those newer customers.



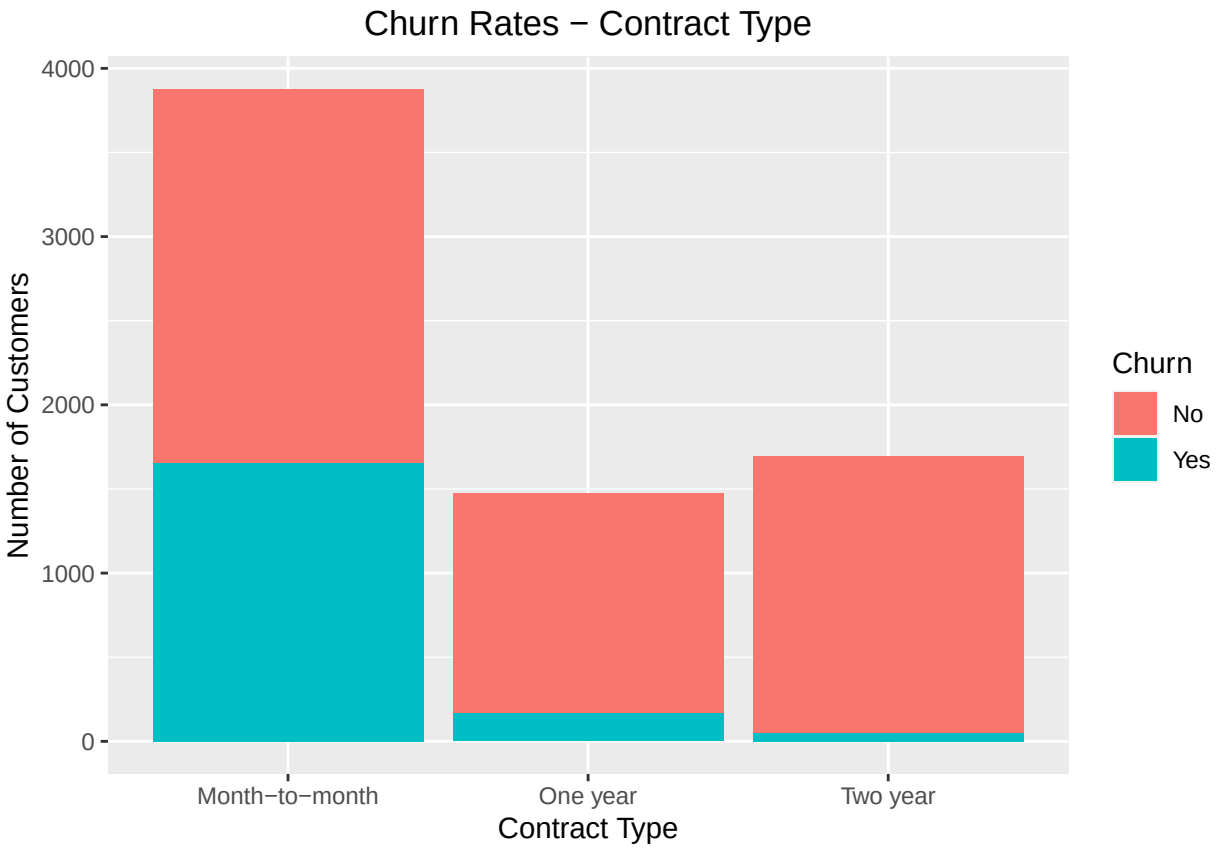
What proportion of customers leave after their first month? An alarming 62%.

```
#more than half of customers with a tenure of 1 are churning
dat %>% filter(tenure == 1) %>% summarize(mean = mean(.$Churn == "Yes")) %>% kable()
```

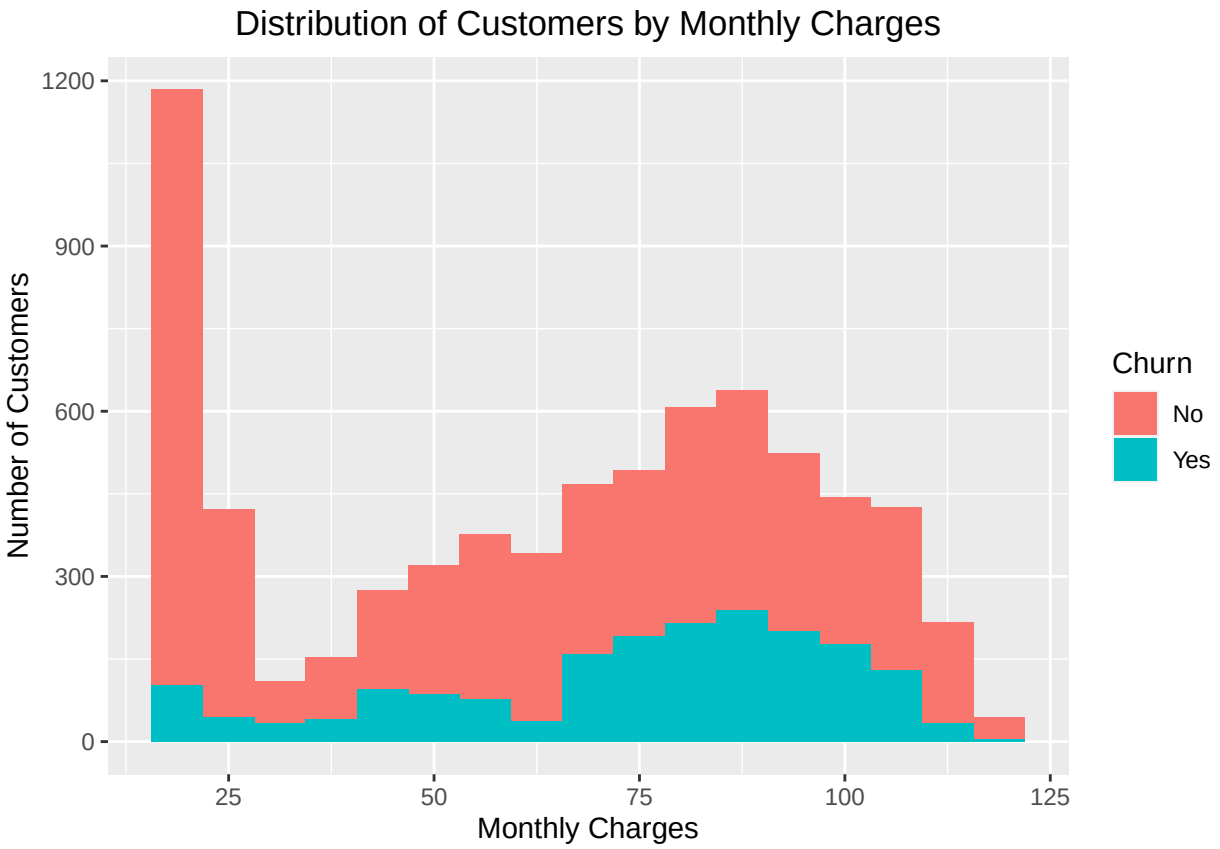
mean
0.6199021

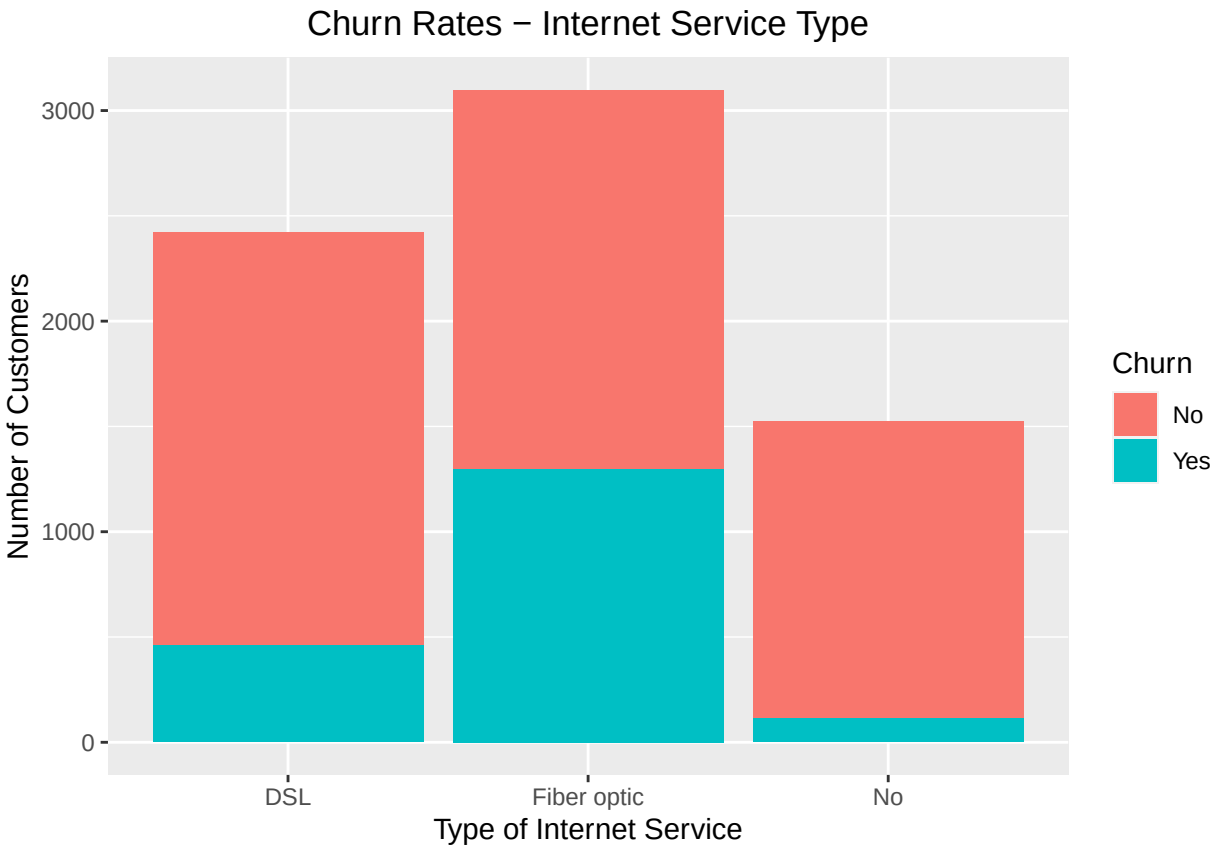
However, there are also some area of strength. Among internet service customers, those who purchase the online security and online backup products churn at lower rates than those who do not. Examine the "Yes" bars below. (Customers represented in the "No internet" bar are not eligible to purchase these products. This unique group is discussed further below.)

The company enjoys an intuitive advantage among customers with longer contracts. These customers churn less.



Customers with lower monthly bills churn less, as do customers who do not buy internet service.





There is considerable overlap between these last two groups. Customers who do not buy internet service are 1526 of the 3460 customers with monthly charges of less than $70.

Methods and Analysis - Part 2: Creating a Model

The client wants to solve a classification problem. Customers fall into two classes: those who churn and those who remain with the company. Large datasets with many variables are a perfect use-case for machine learning with the caret package. The machine learning models described in this report consider all other variables in the dataset as predictive features to predict the classification of a customer's churn status as either "Yes" or "No".

There is only one exception: customerID. Customers are assigned customerIDs randomly to protect their anonymity. The customerID reflects no information about the customer. Additionally, each customerID occurs exactly once in the dataset. Therefore, it is not possible to find any meaningful patterns which involve this variable. The customerID column is withheld when specifying training data for the machine learning models.

The data is split into training and test sets:

```
#Create a test index
test_index <- createDataPartition(dat$Churn, times = 1, p = 0.5, list = FALSE)

#Create training and test sets
train_set <- dat[-test_index,]
test_set <- dat[test_index,]
```

The first model is a random forest model. This popular model often produces good results even without careful tuning. The code below will optimize the model for overall accuracy.

```
#normally this may be set to TRUE so that the user can observe the the work in progress
#it has been set to FALSE to produce a cleaner report
trainctrl <- trainControl(verboseIter = FALSE)

#mtry parameter tells RF how many splits to try at each node
mtry <- expand.grid(mtry = seq(2,10,2))

#fit the RF model
fitRF <- train(Churn ~ .,
               method = "rf",
               data = train_set[,2:21],
               na.action = na.roughfix,
               metric = "Accuracy",
               tuneGrid = mtry,
               trControl = trainctrl)
```

Next the model is used to create predictions on the test set, so that it can be evaluated.

```
y_hat_rf <- predict(fitRF, test_set, na.action = na.roughfix)

#Inspect the confusions matrix to measure the performance of the random forest model
rf_cm <- confusionMatrix(data = y_hat_rf,
                reference = test_set$Churn,
                positive = "Yes")

rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2368  499
##        Yes  219  436
##
##                Accuracy : 0.7961
##                  95% CI : (0.7824, 0.8093)
##     No Information Rate : 0.7345
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.422
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4663
##             Specificity : 0.9153
##          Pos Pred Value : 0.6656
##          Neg Pred Value : 0.8260
##              Prevalence : 0.2655
##          Detection Rate : 0.1238
##    Detection Prevalence : 0.1860
```

```
##       Balanced Accuracy : 0.6908
##
##        'Positive' Class : Yes
##
```

The overall accuracy is approximately %80. This may seem like a great performance, but it is due partly to the imbalanced classes. Many more customers stay ("No" class) than churn ("Yes" class), and the model is better at predicting the "No" class than the "Yes" class. A quick look at the confusion matrix - and the "sensitivity" and "specificity" statistics - makes this apparent. The model is correct about 80% of the time overall, but it only correctly identifies customers who churn about 50% of the time.

The process is repeated with two more machine learning algorithms: k-nearest neighbors (knn) and generalized linear model (glm). The code used to fit these models is quite similar to the code shown for the random forest model, so it is not shown in this PDF report.

Here is the confusion matrix for the KNN model:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2422  608
##        Yes  165  327
##
##                Accuracy : 0.7805
##                  95% CI : (0.7665, 0.7941)
##     No Information Rate : 0.7345
##     P-Value [Acc > NIR] : 1.645e-10
##
##                   Kappa : 0.3369
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.34973
##             Specificity : 0.93622
##          Pos Pred Value : 0.66463
##          Neg Pred Value : 0.79934
##              Prevalence : 0.26547
##          Detection Rate : 0.09284
##    Detection Prevalence : 0.13969
##       Balanced Accuracy : 0.64298
##
##        'Positive' Class : Yes
##
```

Here is the confusion matrix for the GLM model:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2322  429
##        Yes  265  506
##
```

```
##                Accuracy : 0.803
##                  95% CI : (0.7894, 0.816)
##     No Information Rate : 0.7345
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4648
##
##  Mcnemar's Test P-Value : 6.117e-10
##
##             Sensitivity : 0.5412
##             Specificity : 0.8976
##          Pos Pred Value : 0.6563
##          Neg Pred Value : 0.8441
##              Prevalence : 0.2655
##          Detection Rate : 0.1437
##    Detection Prevalence : 0.2189
##       Balanced Accuracy : 0.7194
##
##        'Positive' Class : Yes
##
```

None of these models drastically outshines the others. In fact, they have similar performance characteristics: poor sensitivity and high specificity. However, they do not make the *exact* same predictions. Perhaps performance can be improved by a few percentage points by using them together in an ensemble. This can be done easily with the caretEnsemble package.

```
trainctrl_ensemble <- trainControl(method = "cv",
                                   number = 5,
                                   savePredictions = "final",
                                   classProbs = TRUE)

#tuneList allows us to pass tuning parameters to individual models
#pass the optimal tuning parameters from previous model fittings in this report to reduce computation
tunelist_ensemble <- list(
                          glm = caretModelSpec(method = "glm"),
                          rf = caretModelSpec(method = "rf", tuneGrid = data.frame(.mtry = 4)),
                          knn = caretModelSpec(method = "knn", tuneGrid = data.frame(.k = 29))
                          )

#use tuneList instead of methodList to pass tuning parameters
model_list <- caretList(Churn ~ .,
                        data = na.roughfix(train_set[,2:21]),
                        trControl = trainctrl_ensemble,
                        tuneList = tunelist_ensemble,
                        continue_on_fail = FALSE,
                        preProcess = c("center","scale"))
```

An ensemble will take the majority vote of the three algorithms. When building an ensemble, it makes sense to use algorithms that have some variation in their respective predictions to provide a diversity of predictions in the votes. If the models are too similar to each other, they all vote the same way and there is nothing gained by using multiple models instead of just one. The caretEnsemble package provides a simple way to inspect the correlation of the models in the ensemble with the modelCor() function.

```
modelCor(resamples(model_list)) %>% kable()
```

	glm	rf	knn
glm	1.0000000	0.7376983	0.6433637
rf	0.7376983	1.0000000	0.1221964
knn	0.6433637	0.1221964	1.0000000

This is not excellent diversity, but the ensemble will be built anyway.

```
ensemble <- caretEnsemble(
  model_list,
  metric="ROC",
  trControl=trainControl(
    number=2,
    summaryFunction=twoClassSummary,
    classProbs=TRUE
  ))

#Make predictions using the ensemble
y_hat_ensemble <- predict(ensemble, test_set, na.action = na.roughfix)
```

Here is the confusion matrix for the ensemble predictions:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2337  444
##        Yes  250  491
##
##                Accuracy : 0.803
##                  95% CI : (0.7894, 0.816)
##     No Information Rate : 0.7345
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4589
##
##  Mcnemar's Test P-Value : 2.368e-13
##
##             Sensitivity : 0.5251
##             Specificity : 0.9034
##          Pos Pred Value : 0.6626
##          Neg Pred Value : 0.8403
##              Prevalence : 0.2655
##          Detection Rate : 0.1394
##    Detection Prevalence : 0.2104
##       Balanced Accuracy : 0.7142
##
##        'Positive' Class : Yes
##
```

Overall, the ensemble performs similarly to the individual models.

Results

The machine learning approach performs well enough to provide useful information to the company. It outperforms the theoretical naive approach of simply guessing the mode classification for every sample. The main shortcoming is sensitivity. The best models only identify customers who will churn about half the time. However, it may be that there is simply not useful information among the predictive features to improve sensitivity further. The table below summarizes the model results. For comparison, the naive approach is also shown.

Model	Accuracy	Sensitivity
Naive Approach	0.7346301	0.0000000
Random Forest	0.7961386	0.4663102
K-Nearest Neighbors	0.7805224	0.3497326
Generalized Liner Model	0.8029529	0.5411765
Ensemble	0.8029529	0.5251337

The GLM and ensemble models stand out as the best performers. The random forest model has similar accuracy, but it falls behind in sensitivity, which is an important metric for this situation. (It is important to detect the customers who will churn so that the company can attempt to retain them.) It is worth noting that the GLM model can be built in a tiny fraction of the time it takes to build the ensemble. For this reason, the GLM model is a clear winner. It could practically be scaled to handle millions of observations - perhaps the company's entire set of customer records. The random forest, k-nearest neighbors, and ensemble models are too computationally intensive to scale easily.

Conclusion

This project aimed to predict customer churn at a telecommunications company in order to provide that company with useful business intelligence. Several approaches were tested, and the best model yielded an overall accuracy of approximately 80% and sensitivity to churning of about 50-55%.

Uncertainty is the norm in business, and conditions do not remain stable long enough to allow the development of perfect forecasting models. The model is accurate enough that it can serve as a useful tool for predicting future revenue when combined with other information (predictions about how many new customers will be added, predictions about changing prices for TV content, etc.).

In broad data science terms, a sensitivity of 0.50 - 0.55 is not stellar. In real-world terms, the model has successfully flagged half of the customers who are going to leave next month, in addition to a handful of false positives (the positive prediction value is around 65%). This is highly actionable information, because this is a relatively small group of customers who can be targeted with retention offers that should not be offered to the broader population, such as discounts or bonus services that incentivize staying with the company. At the very least, the model provides a short list of customers worth reaching out to.

The data exploration yielded some insights that merit further investigation. Why are so many customers trying the company's service for one month, and then leaving? Is there something the company could do better in order to retain them? Why do the online security and online backup features seem to associated with customer loyalty? Why do customers with higher bills churn more frequently? This project lacks the tools to answer these questions here and now. However, the data provides the company with a useful starting point for further research.

The greatest shortcoming of this project was the middling sensitivity achieved by the best model. Future efforts should explore additional models with the goal of achieving higher sensitivity. Future efforts should also consider reworking the ensemble to include models with similar accuracy but lower correlation.