Data Mining Application in Telecommunication Churn Management

Gavin Mccullion

Chuxin Xue

Omkar Kulkarni

Table of Contents

Executive Summary

The goal of this project was to identify which customers would unsubscribe (churn) from a regional cell phone company. The telecom industry has been an area with a large amount of analytic research dedicated to customer retention, due to the large amounts of data generated and the saturation of the marketplace. Customer retention is five to six times cheaper than attracting new customers, and a one percent increase in retention can raise share prices by five percent.

Our data was sourced from a regional cell phone company located in the Southeastern United States, and contains customer information from 1996 through 1999. The overall churn rate for the customer base was 37%. Certain demographics had notably higher or lower churn rates. For instance, customers far from the southeastern region churned at very high rates (70% +).

After initial exploration of some of our data and how it related to customer churn rates, we added variables to explore groups of customers not readily identified by categories in the original data set. One group of customers was those with any level of outstanding billing. Another group was those customers who appeared to be with the company for a longer period of time. This group was identified by having a renewal date different from their activation date. Two variables were made to describe these customers – one was their time as a customer, in weeks, the other was a variable to distinguish them from new customers.

Graphical analysis and initial modeling revealed that the group of customers we identified was churning at approximately 95% rate. Identifying customers likely to churn pointed to these customers overwhelmingly in our models, so we created two data sets – one which kept the identifying variables for the old customers, and one which removed it.

Several different data mining algorithms were applied to find which model would yield the greatest prediction accuracy. A decision tree, was initially promising, scoring 95% on the data set with the extra variables and 77% on the new data set. Decision trees split the data on important variables – for instance, if a customer has outstanding debt, he will go into one category, if he has no debt, he will go into another. This process is repeated through several variables until an outcome is reached.

The best model for predicting churn was a random forest, which takes the concept of a decision tree and improves upon it by creating dozens (or hundreds) of possible decision trees, then aggregating them to find the best fit. This model yielded 99% accuracy on the data set with the extra variables, and 83% on the data set with those variables removed.

Now that our company is able to identify customers likely to churn, retention strategies can be deployed in order to keep customers renewing their contracts. Such strategies can include phone upgrades, service upgrades, and flexible service plans.

Future data collection will allow us to better target customers, especially if we can determine reasons for churning. Further model evaluation and optimization will also be investigated on future versions.

1.1 Business Understanding

The telecom industry has been an industry with great potential for data miners to apply their knowledge to. Telecom companies generate large amounts of data, and face stiff competition, especially in a saturated marketplace where there are few truly "new" consumers purchasing for the first time. Because of this level of competition, companies always prefer retaining customers over attracting new ones. Retaining customers is five to six times cheaper than attracting new ones[1]. Thanks to the availability of large amounts of data, companies are able to utilize data mining to reduce churn, which is the measure of how many customers leave a company.

Some reasons customers may decide to churn include[2]:

- Poor network coverage/reception quality
- Plan Pricing
- Inadequate features
- Poor customer service and billing errors

Many different algorithms have been used to predict churn for the telecom industry. Studies utilizing neural networks[3] and support vector machines[4] yield high accuracy rates near 90% for predicting churn. Simpler methods are often used as well – decision trees are the most frequently used method in predicting churn[5].

2.1 Data Understanding

In order to do a comprehensive data exploratory, we used Tableau to visualize our data, and also the Data.Table package in R to summarize result by group.

Churn Rates

The overall churn rate for the entire data set is 37%. This is a rate much higher than the big four telecom companies see in the present day, however for a small, regional company from fifteen years ago, this can be expected. Since churn is what we are trying to reduce, we examined churn rates for the variables within the data set (at this point unmodified – no new variables or data cleaning) to determine if there are any demographics which stand out as high churners. Continuous variables were used to create new variables, which will be discussed under data preparation.

---

[1] Almana, Aksoy and Alzahran. "A Survey on Data Mining Techniques In Customer Churn Analysis for the Telecom Industry." http://www.academia.edu/7675641/A_Survey_On_Data_Mining_Techniques_In_Customer_Churn_Analysis_For_Telecom_Industry.

[2] Almana, Aksoy and Alzahran. "A Survey on Data Mining Techniques In Customer Churn Analysis for the Telecom Industry." http://www.academia.edu/7675641/A_Survey_On_Data_Mining_Techniques_In_Customer_Churn_Analysis_For_Telecom_Industry.

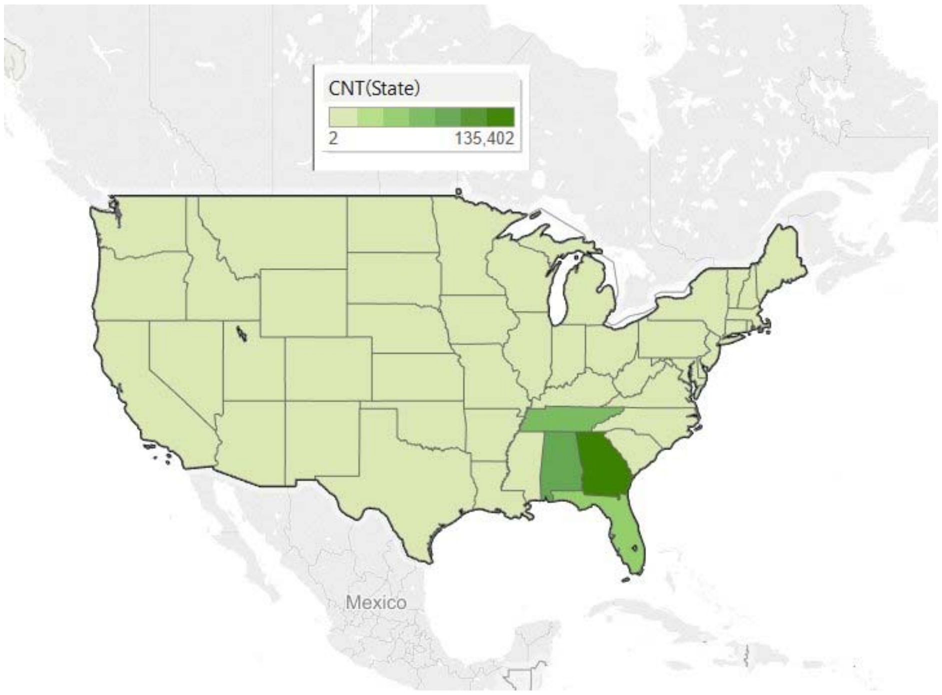
[3] Sharma and Panigrahi. "A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services ." http://arxiv.org/ftp/arxiv/papers/1309/1309.3945.pdf

[4] Brandusoiu and Toderean. "CHURN PREDICTION IN THE TELECOMMUNICATIONS SECTOR USING SUPPORT VECTOR MACHINES." http://imtuoradea.ro/conf/2013/Brandusoiu%20Ionut%201.pdf

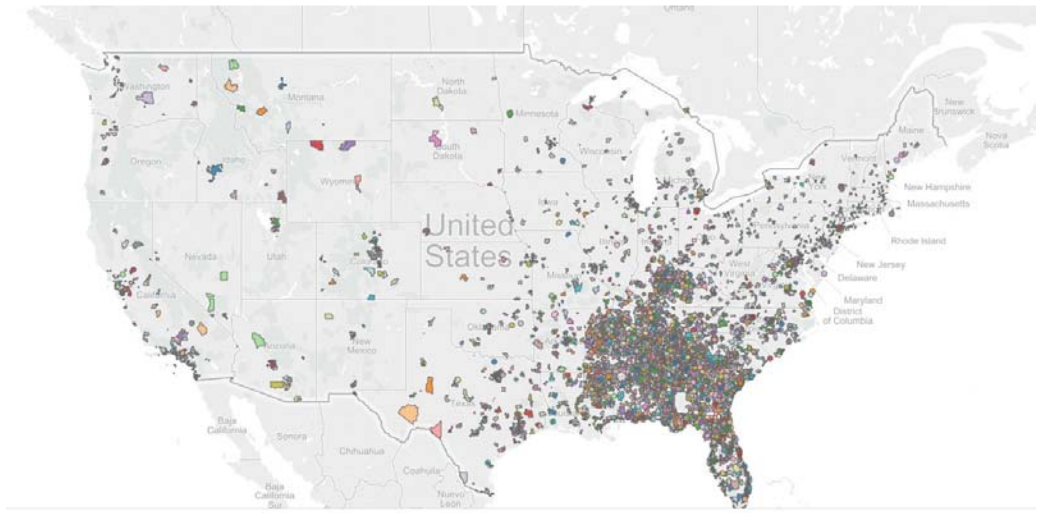
[5] Almana, Aksoy and Alzahran. "A Survey on Data Mining Techniques In Customer Churn Analysis for the Telecom Industry." http://www.academia.edu/7675641/A_Survey_On_Data_Mining_Techniques_In_Customer_Churn_Analysis_For_Telecom_Industry.

Customer Location

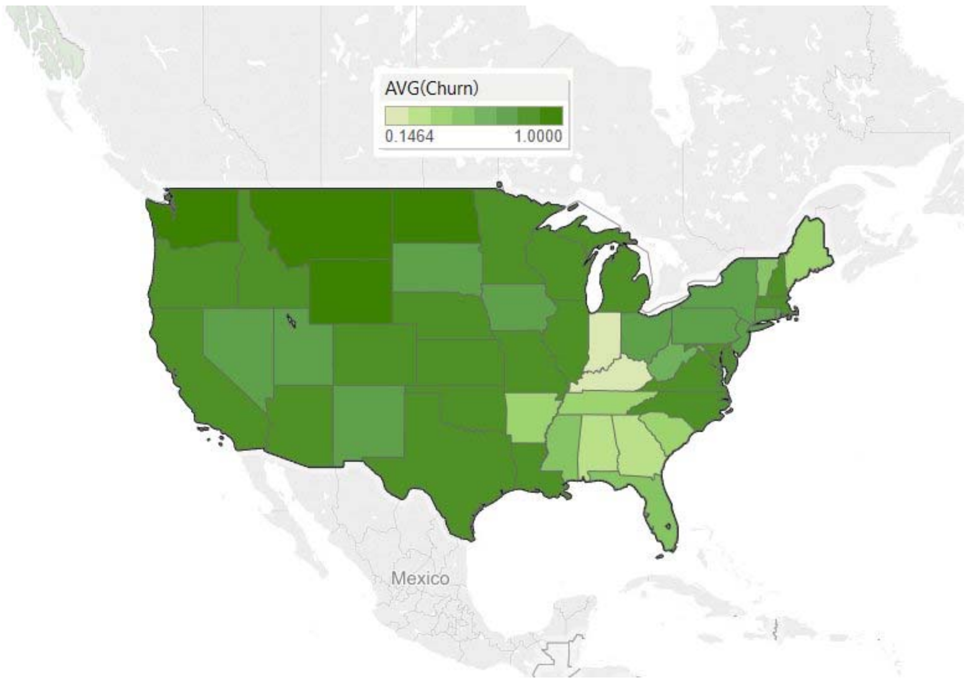
Customers of our telecom company's dataset are in all fifty states, however when we look at the count of customers by region we can see that the company is almost exclusively regional, as most customers are located in the southeastern portion of the country. A heat map of customers per state illustrates this well:



Plotting the zip codes where customers are located is perhaps even more illustrative:



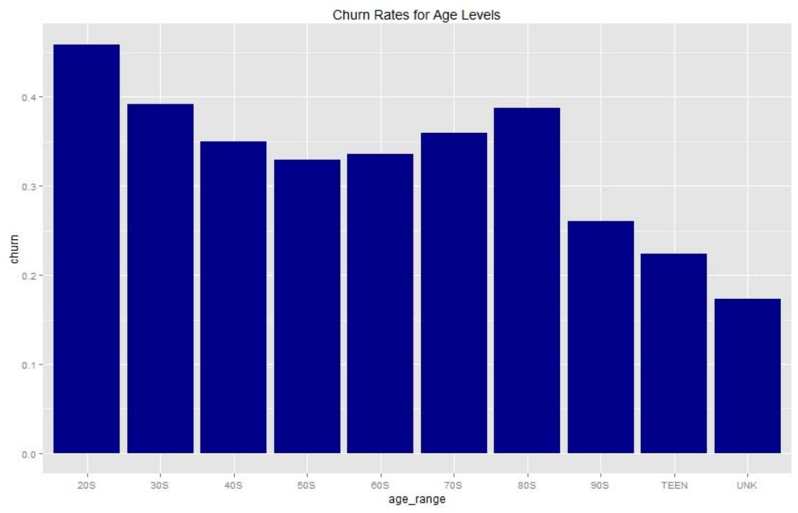
If we look at the churn rates for the different states and regions, it's practically the inverse of our population heat maps. When we start to get outside of the southeastern region, churn rates shoot up incredibly high.



This is likely due to customers moving away from the southeast and finding less network coverage, and possibly roaming charges for being outside the region. Kentucky and Indiana seem to be immune from the higher churn rates, probably because customers there do not have network or roaming issues.

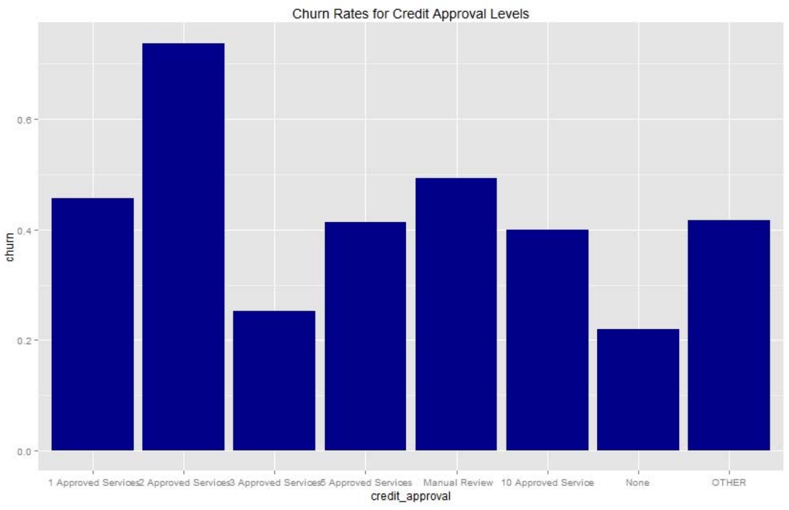
Age

When we look at churn rates by age, there's not a clear linear pattern. Aside from those customers where age is unknown, teens have the lowest rate (likely that they are on the same carrier as their parents, and do not get to choose whether or not to churn), and those in their 20s have the highest. Churn rates decrease until we reach customers in their 60s, at which point it rises again. One possibility is that customers have busier lives and more obligations to deal with, but upon retirement have more time to consider changing cell phone providers.



Credit approval

There are five levels of approved credit, along with manual review, none and other. We do not know how the levels of approval map to credit scores, but a look at the churn rates shows that the 2 Approved level has the highest rate, with 3 Approved having the lowest. None/Other for credit approval may be for business customers.

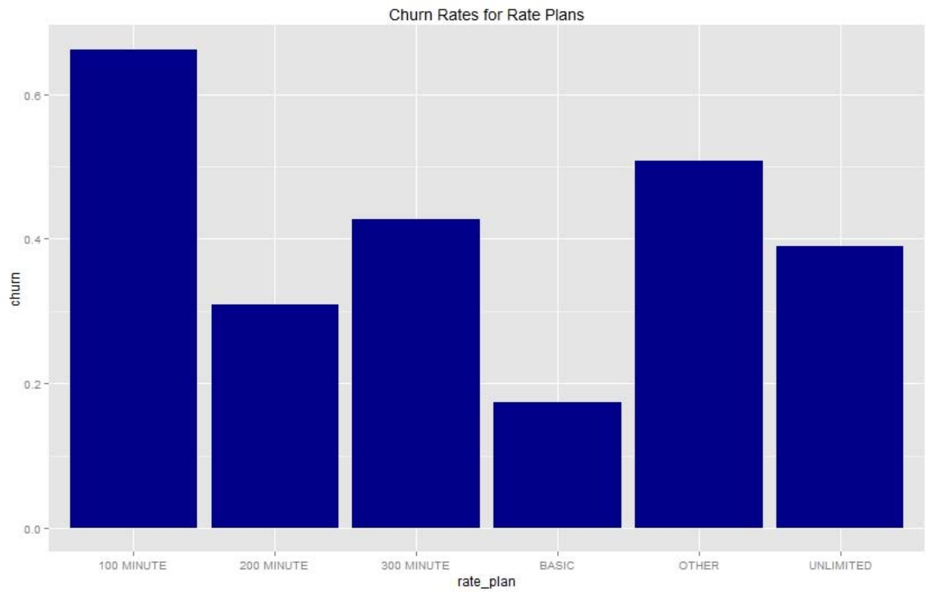


Contact Method

Those who purchased their phones/service plans at mall kiosks or over the internet had churn rates of over 50% (0.53 and 0.56 respectively. Those who are coded as family sale, retail sale, and other all had churn rates within a few percentage points of the average 37%. Phone sales had noticeably low churn rates, only 19%.

Rate Plan

The 100-minute plan and any customers coded as 'Other' had the highest churn rates, at 66% for the 100-minute plan and 51% for Other. The 200 Minute, 300 Minute and Unlimited plan had rates that did not deviate too far from 37%. Basic plan subscribers had a very low churn rate, at only 17%.



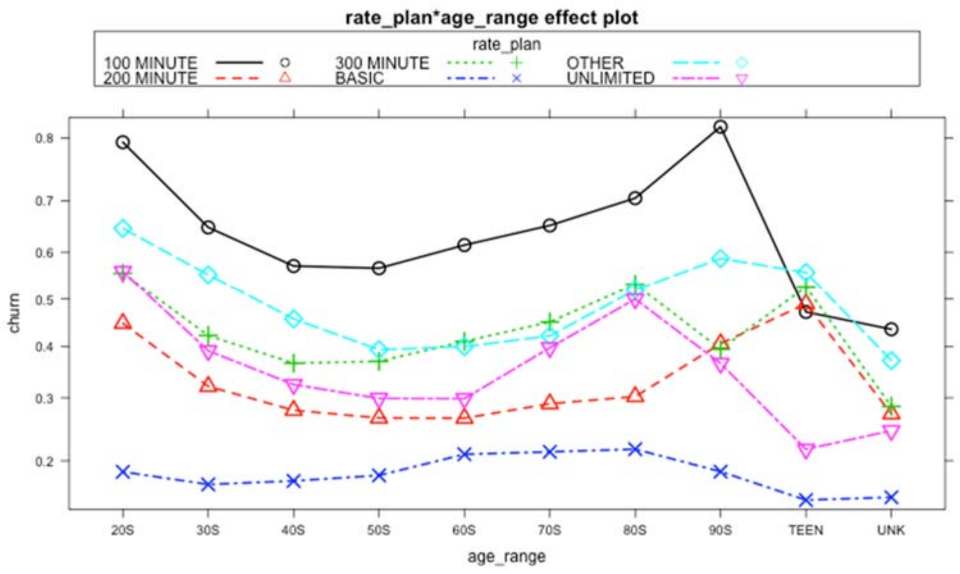
3.1 Data Preparation

Outlier Detection

Though we only have four numeric variables in our dataset, we think it's still necessary to check outliers because large dataset like ours are more likely to suffer from outlier issue. We visualized the distribution of our numeric variables and found that all of them are highly left-skewed. However, no outliers are observed in the distribution, therefore we didn't remove any observations.

Interaction Effect

We are also interested in understanding the interaction effects between variables in the data as the number of levels presented in some of our variables is quite high. Looking into their interaction will help us further interpret the result especially in logistic regression. The package we used is called effect. We did analysis on four pairs of variables, and plot each of them to visualize their interaction. Three of the pairs are found to have interaction effect and will be added to the logistic regression model. Below is an example plot from our result.



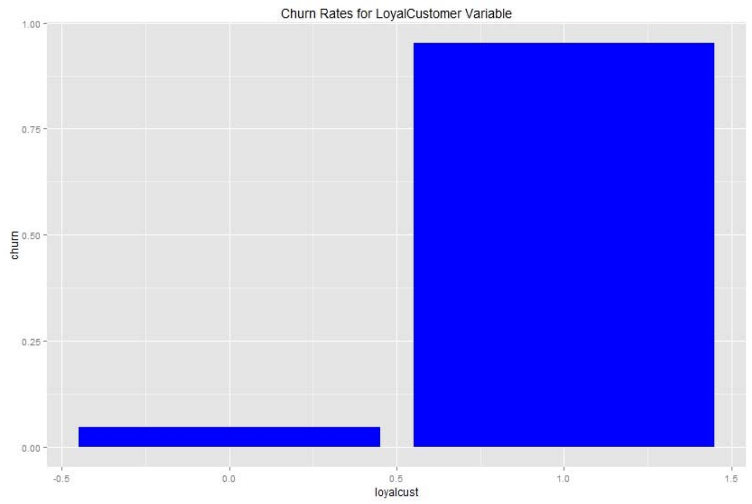
Variable Removal

Based on our examination of the data, we determined that specific user location data would not be necessary for our analysis. Since we already have a variable for region, this would suffice for describing customer location in our models. Furthermore, the amount of levels within the city and zip code variables would be problematic when running models on a data set of this size. We also removed the customer id variable, as this would only add noise to our models.

Variable Manipulation (Original dataset)

Customer activation date and renewal date were formatted in the original data set as one column for the year and one column for the month. In order to use these in our model, we created a function to change the dates into a month/day/year format. Since we only have month and year data, we set the day equal to the first of the month, i.e. June of 1999 became 6/1/1999.

About one-third of the records in the dataset had different renewal and activation dates. Without a data dictionary, we assumed that these were the customers that had already renewed their service agreements. We believed that it was important to capture how long a customer was with the company, so we took the number of weeks between the activation and renewal dates and



created a variable titled customerlifetime to capture this data. We also wanted to capture customers who had already renewed versus new customers. A new variable, loyalcustomer, was created. For any customer with a customerlifetime greater than zero (which is to say, their activation date and renewal date was different), loyalcustomer was coded as a 1. All others received a value of 0.

In examining churn rates for our ‘loyal customers,’ our variable name proved to be rather ironic. For those with a 1 in the loyalcustomer variable, there was over a 90% churn rate. These two variables, loyalcustomer and customerlifetime, would prove to be very important in our modeling. Running our models with and without these variables greatly changed the accuracy.

Another variable we created was a binary variable called debt, which captures any customer who had a total_open_amt greater than zero. Those with a debt variable equal to 1 had higher churn rates than those who did not have debt.

Two Data Sets

Preliminary modeling showed that the loyalcustomer and customerlifetime variables dominated our models. Because of the possibility of leakage resulting from these variables, we created two separate data sets to work with: one which includes the new variables, and one which removes customerlifetime, loyalcustomer, renewal_data and activation_date.

Finally, we split the data into train, test, and validate sets. We did this by creating indexes choosing random observation numbers from the datasets. These subsets were split into 70%/20%/10% of the original data sets.

4.1 Modeling

Decision Tree

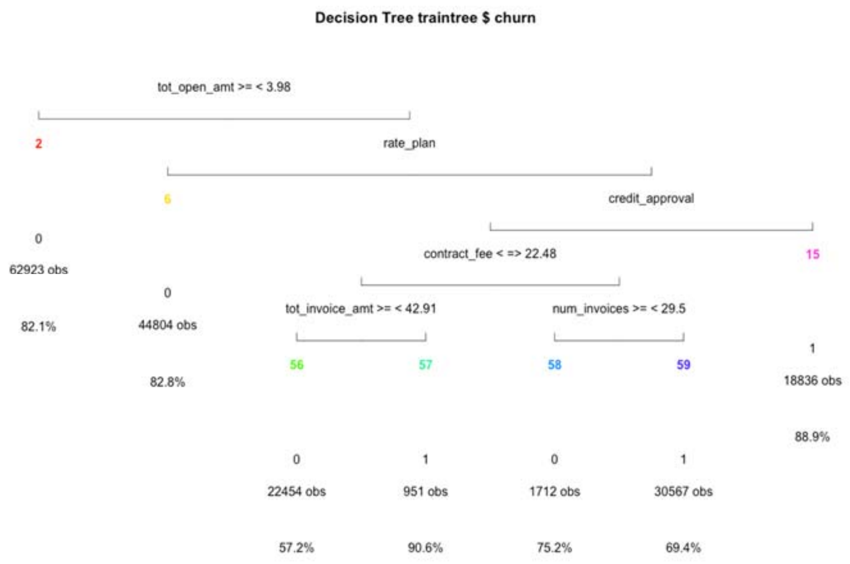
The first method we tried is decision tree because it's considered as the most easy-to-interpret method and also works very well with categorical dataset. With a large dataset that has 12 categorical variables, decision tree should be an ideal method.

Data Preparation

To prepare the data, we first checked the class of all variables and found some categorical variables are classified as numeric. We converted them back to factors, as this would help the model run more smoothly.

Model Building

We started off growing the tree using 'rpart' to test what parameters will yield the lowest cross-validation error rate. The CP table and plot suggested that the best CP is around 0.01 and the best number of split is 5. We then fine tune the model with these parameters in the rpart control . As we were not very sure about how the minimum split and minbucket will affect our model, we decided to try out a few combinations. We tested the model again with a few low and high values of minsplit and set the minbucket to be 1/3 of minsplit(as best practice). Interestingly the accuracy of the train model stayed the same.



Though accuracy is our number one criteria when evaluating a model, we also want to make sure variables the model selected are reasonable and interpretable.

For the original dataset with dates, the tree model only selected three variables (renewal_date, customer lifetime, number of invoice), all of which are all related to the length of time the customers has been with the company. Loyal customers are far more likely to be churners. The final accuracy on test yields around 95%.

As to the our new dataset (without dates), the tree plot revealed that five variables were selected in the order of importance – rate_plan, contract_fee, credit_approval, tot_open_amt, and number of invoice. Based on the tree, we may be able to conclude that customers who have unpaid balance, fewer credit approval services, and stay with the company for a long time (more than 29 invoices) are more likely to be churners.

Much as we expected, the best accuracy the tree model for the new dataset yielded only 77.4% on train dataset and 77.3% on test dataset.

Support vector machine

Data preparation

As support vector machine is a classification algorithm that works with multi-dimension space, we first had to transform the data into numeric. It was quite a challenge for us because most of our variables are categorical. In order to convert the data into numeric, we had to create dummy variables for each level at each categorical variable. Considering this process would only complicate our dataset making it too large to handle, we have collapsed levels for some variables to reduce the dataset dimension.

The rules we used to combine levels are demonstrated in the neural network section. After the collapsing, we went on to create dummy variables for each new level and were able to come up with a pure numeric dataset for support vector machine.

Model building

Due to the complexity of the algorithm and our dataset size, the model building process of support vector machine was the most time-consuming of all. The results are very different for the two datasets. The package 'Kernlab' is what we used in the project. At the beginning, we used the original dataset (which has the dates variables) and successfully ran a few rounds using 'linear', 'poly' and 'rbf' methods after several hours' waiting. We found that the 'rbf' method gave us the highest accuracy of 95%, followed by poly and then linear. This seems reasonable because rbf handles non-linear classification, which usually works well with dataset with many attributes.

However, when we tried to run svm on the new dataset (without dates), things are not as smooth as we thought. The model took more than one night to run and never got us any valid results. It may due to the fact that removing those suspicious leakage variables have made the data pattern harder to interpret. Therefore the algorithm will need a lot more time to search for solutions.

In the interest of time, we decided to down size the dataset by sampling only 1% of the train data and do a prediction on 1% of test data to see what results we can get. We ran the model using all three methods (rbf, linear and poly) and wrote a for-loop for each method to test out different sets of parameters. Interestingly, we found that linear method seems to work slightly better than poly and much better than rbf. By comparing the 5-fold cross validation errors from the loop results, we identified the best C is 1. With our best model, we ran it on four different 1% random samples from the test data. The accuracy ranges from 75% to 77%. We consider these as our final result for svm.

Random Forest

Random Forest has always been a reliable algorithm. Though it's based on decision tree, RF provides higher accuracy because it can overcome the over fitting issue by randomly sampling predictors at each split. But due to its complexity, we had to set up more parameters. Number of trees and mtry are the two main ones. To improve efficiency, we used the tuneRF function in the randomForest package to help us find the best combination of ntree and mtry values. Limited by time, we only got the chance to run two scenarios. First , ntreeTry was set to 100 with a stepFacotr of 1.5 and improve of 0.01. We found that when ntree=100, mtry=3, the accuracy is the highest. We tried again with an ntreeTry of 500, interestingly, the result is also optimal when mtry=3.

A further test was done to evaluate their predictive power on new data. Both the 100-tree-model and the 500-tree-model gave us very similar accuracy results. Their out-of-bag accuracy is between 82-83%, accuracy on train data is around 93%, and accuracy on test data is almost exactly the same as their OOB value. We looked up the definition of OOB error and realized that it's actually Random Forest's own version of cross validation, which is estimated during the tree building process. Therefore, we are more confident with our result. The 500-tree model with 3 mtry yields the lowest error rate of 17.01% (82.9% accuracy)on the test dataset.

Boosting

Another algorithm similar to decision tree is boosting. Adaptive boosting is the method we focused on. At first we used a package called gbm, which is often recommended for boosting. We tested it out with n.trees of 150 and shrinkages of 0.1, and the summary result revealed that only four predictors are used by the model– tot_open_amt, credit_approval, rate_plan and tot_invoice_amt, which is consistent with what we saw in the tree result.

Var	**Rel.inf**
tot_open_amt	39.9083382
credit_approval	31.2209424
rate_plan	28.7700513
tot_invoice_amt	0.1006681

However, with the gbm package, we are only given the probabilities of each observation but not the predicted responses. We spent some time reading the manual but still couldn't find the way to convert it back to responses and calculate the accuracy. We then did more research and decided to try out the package 'caret', which is proved to be very helpful in searching optimal parameter for us based on the ROC criteria. The caret result recommended us to use an iteration of 150 and a nu of 0.1 as our parameters.

To run the model with the best parameters we found, we used the 'ada' package and got an out-of-bag error of 19.6%. On the test data, the final accuracy is 80.4%.

Neural Network

The use of neural networks for modeling data requires two things: numerical data, and standardized data. Our dataset is largely composed of categorical variables, which needed to be converted to numeric equivalents before training the neural network.

The categorical variables were not ordinal, so it was not feasible to simply replace a five-level variable with numbers 1 through 5. Instead, dummy variables were created, where each level of a variable becomes its own, separate variable. For instance, for credit ratings, '1 approved' becomes its own variable, with a value of 1 for all customers who have a '1 approved' credit, and 0 for all other customers. This creates a large number of new input variables, which is an issue since neural networks need a relatively low number of input variables to work well.

New Variables	Level	Churn
credit_approval		
cahigh	2 approved services	0.73
camid	1 approved services	0.45
	5 approved services	0.41
	Manual review	0.49
	other	0.41
calow	3 approved services	0.25
	none	0.22
contact_method		
contact_low	PHONE SALE	0.19
contact_mid	FAMILY SALE	0.38
	OTHER	0.34
	RETAIL SALE	0.35
contact_high	INTERNET SALE	0.56
	MALL SALE	0.53
rate_plan		
rate_low	BASIC	0.17
rate_mid	200 MINUTE	0.31
	300 MINUTE	0.43
	UNLIMITED	0.39
rate_high	100 MINUTE	0.66
	OTHER	0.51

To solve this, dummy variables were combined where variable levels had similar characteristics. For the region variables, levels were combined based on subscriber levels. E_S_Central and S_Atlantic were the two largest regions, combined as region1. W_S_Central and E_N_Central had a few thousand subscribers each, so they became region2. All others had a few hundred subscribers at most, and were outside the core region. These remaining regions were combined as region3.

For our other variables, churn rates were used to combine levels into dummy variables. Age range was left out as our decision tree model showed that it was not pertinent to predicting churn.
Different numbers of hidden layers were tested to discover what the optimal fit for our neural network. Results are below for both the original dataset (with dates) and new dataset (without dates).

Table 1- NN Layers of Original Data

Layers	Error Rate
1	0.0795
2	0.0516
3	**0.0458**
4	0.047
5	0.0464
6	0.0783

Table 2 - NN Layers of New Data

Layers	Error Rate
1	0.21
2	0.207
3	0.202
4	0.204
5	0.1997
6	0.1996
7	0.197
8	0.195
9	**0.193**
10	0.195

When training our dataset that excluded our ‘loyalcustomer’ and ‘customerlifetime’ variables, a larger number of layers gave the lowest error rate. There is some concern that a large number of layers can lead to overfitting, and the accuracy difference between 3 hidden layers and 9 hidden layers is only 0.9%. Had this been chosen as our model to deploy, we would likely choose the lower number of hidden layers.

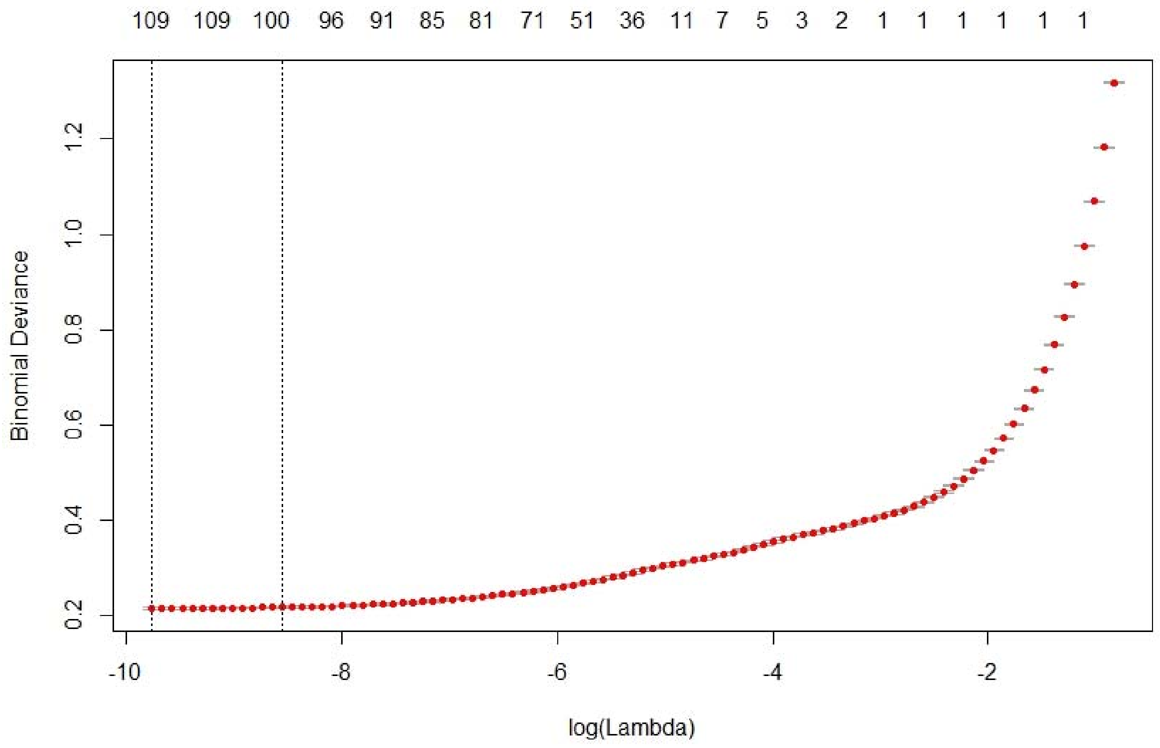
Various levels of variable weight decay were attempted, but keeping decay at the default value of 0 proved to be the optimal setting.

Cross validation was attempted with our neural network using the tune.nnet procedure from the e1071 package. This procedure uses k folds cross validation to find the optimal setting for a neural network. Unfortunately, due to the size of the data set and available computing power, this process could not be run.

Logistic Regression

When using our dataset with the loyalcustomer and customerlifetime variables, running a logistic regression using the glm procedure from base R yields the following error: fitted probabilities numerically 0 or 1 occurred, algorithm did not converge. This is a case of linear separation caused by our customerlifetime variable separating the data perfectly. In order to remedy this issue, a form of penalized regression was employed.

The glmnet package in R creates penalized regressions based on maximum likelihood estimates. The penalty parameter lambda is denoted in the code with option ‘s.’ At a 0.005 level, we have a 5% error rate. Glmnet includes a method for cross-validating our model fit, so this method was employed to determine the optimal penalization parameter s.



The minimum lambda value is 0.00006. Plugging this value into our model, we lower our error rate to 3%.

When using logistic regression on our dataset that excludes the loyalcustomer and customerlifetime variables, we are able to use the glm function from base R to create a model. A stepwise regression reveals that all input variables should be kept aside from tot_paid_amt. Creating a new regression model with tot_paid_amt excluded gives an error rate of 21.6%.

We can calculate the variance inflation factors using the car package in R. Cutoff points for high VIF can differ based on individual models and the opinions of whomever is creating the model. With a cutoff point of five, two variables exceed the threshold: rate_plan and contact_method. The value of both variables is below ten, which can be considered a more liberal cutoff point. VIF is not always a great way to judge multicollinearity for categorical variables with several factors, which both of our suspect variables happen to be. As such, both rate_plan and contact_method were kept in the model.

Penalized logistic regression with glmnet was also employed for the dataset that excluded the loyalcustomer and customerlifetime variables. The error rate is 21.6% for guessing the s value at 0.005, and 21.1% for our minimum s value of s (0.0027).

5.1 Evaluation

	Random Forest	Boosting	NN	Logit	Tree	SVM
Test data	17.0%	19.6%	20.0%	21.1%	22.8%	22.5%
Validate data	16.9%	19.2%	18.9%	21.3%	22.40%	22.80%

The table above showed us the overall error rates of our models, but before we conclude the best among all models, a very important step is to further evaluate the model results and find out how well they will respond to new dataset. To gain a comprehensive picture of the model performance, we have utilized several criteria to assess our top models. We'll focus on the new dataset (without dates) in this evaluation.

Cross-validation

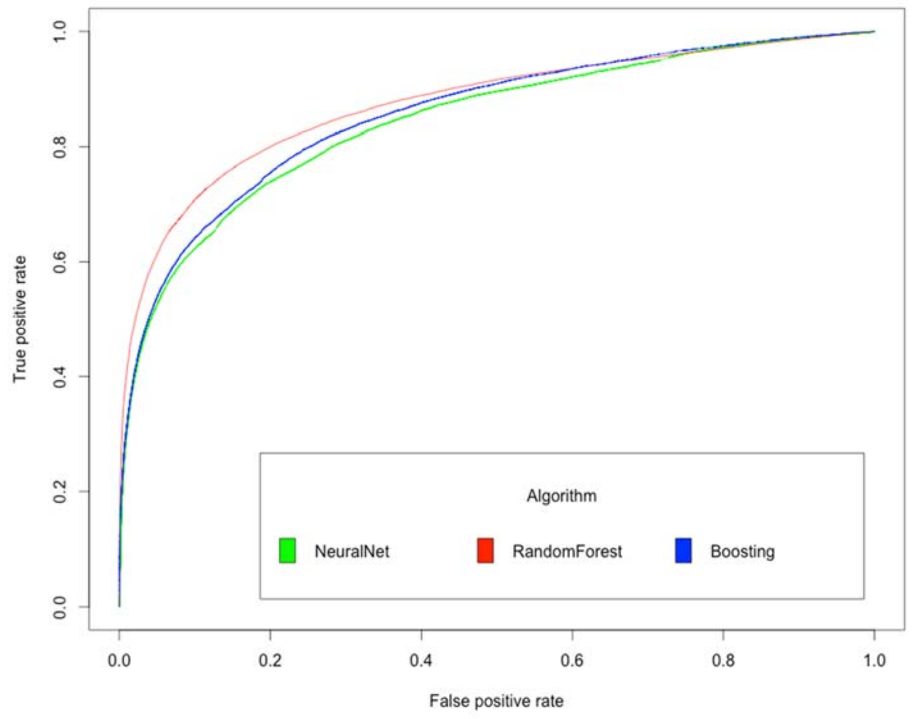
In our model building process for most techniques, we were able to run 5-fold or 10-fold cross validation which gave us an accurate estimation of the model's performance on new data. Cross-validation is especially useful when we were testing out parameters in loops. We were able to pick out the best parameter sets based on the cross validation result from each loop. As expected, the error rate on the test dataset is very close to the in sample cross-validation further proving the reliability of cross validation.

Confusion Matrix

Another method to evaluate a model's performance is through analyzing the confusion matrix. Ideally we want to balance both Precision and Recall of a model to avoid biased prediction influenced by the number of false positives and false negatives, therefore we have calculated F-measure for our top models below. It's obvious that the random forest model has the highest precision, recall, and F-measure among the three.

Random Forest					
	0	1		P	0.686
0	42746	8692		r	0.826
1	3987	18961		F measure	0.749
Boosting					
	0	1		P	0.641
0	42097	9932		r	0.793
1	4636	17721		F measure	0.709
Neural Network					
	0	1		P	0.610
0	42531	10781		r	0.801
1	4202	16872		F measure	0.693

ROC Curve



The last criteria we used to evaluate our models is ROC curve, which provides us a visual comparison of our models based on how accurately they can predict true positives. The perfect classifying model should have a curve along the left-hand border and top border. In the ROC graph above for our top three models, random forest, which is represented by the red line, is the closest to the top left corner indicating that it has accurately predicted the most true positives and the least false positives.

All criteria above have proven that random forest is the best algorithm for our dataset. Though model accuracy is the most important goal of our project, we also find random forest to be an easy technique to work with. The fact that the model is based on decision tree has made it easier to interpret to upper management. Also, it runs efficiently on our dataset because random forest requires very little data transformation and works very well with categorical data.

6.1 Deployment/Recommendations

Deployment

Referring to our decision tree provides a simple method for determining the groups most likely to churn so that business practices can be changed in order to keep those customers on board. Trees provide a simple and intuitive way to describe which groups are churning to any interested party regardless of prior statistical knowledge. For the business of predicting future churn, our random forest algorithm would still be used, as it takes the idea of our decision tree and improves upon its accuracy.

In practice, our algorithm will need to be re-trained and run on new data as time goes on and customer behavior and business practices change. Those responsible for customer retention in the company should be given names of customers most likely to churn, in order to target them specifically with offers to stay with the company. Reporting should be used to check on the accuracy of future iterations of our model, along with determining its efficacy by examining future churn rates.

Recommendations

For the dataset that includes the customerlifetime variable, we find that customerlifetime and loyalcust variables dominate the models. This can be interpreted as our company having poor retention in general, as customers who renew end up churning at a phenomenal rate. Customers whose contracts are coming up for renewal should be incentivized to remain as subscribers. Some strategies to retain these customers include:

- New phones – free upgrades for basic models, subsidized upgrade pricing for newer and cutting edge phones. Our company can also implement a buy back policy for the customer's old handset.
- Free upgrades – customers could receive upgrades to the next tier of service (e.g. 200 minute call plan customers are upgraded to 300 minutes) for a pre-determined period.
- Bonus features – customers without unlimited texting could earn 100 free text messages as an incentive for renewing
- Flexible Plans – customers would be able to customize parts of their plan to fit their individual needs. Certain segments of customers would want to be able to travel internationally and still use their phone without additional charges, others would be interested in texting more than making phone calls, etc.

When we remove the customerlifetime and loyalcustomer variables from our data, the new data set does not give us as clear of a definition as to who is churning. Variables such as debt and credit_approval show up as significant. Unfortunately, we do not have access to the reasons that customers have churned. For instance, knowing exactly why so many customers categorized as "2 Approved" for their credit score have churned compared to all other levels.

Future Study

Further data collection would greatly increase our ability to target customers, as knowing the top reasons for churning would allow us to develop a retention strategy that would appeal more to the customers. Although the goal for this study was overall accuracy, future modeling could examine false positive and false negative rates, and determine if it would be more profitable to focus efforts on reducing one of those in addition to achieving a high overall accuracy rate. Future data collection will also need to determine if the overall dataset has any sources of leakage, and remove those sources before creating models.

7.1 Key Learning Points

Several main learning of this project includes

- Data cleaning is not a one-time process

  Before this project, we thought data cleaning is only a one-time process before the model building and we'll use the same clean data for all the models. However, we have seen that data cleaning should be done on both the overall level and the model level. Different algorithms has their unique way of interpreting data pattern, therefore it's essential for us to try out different ways to clean the data and compare results after running the models.

- Cross validation is extremely important

  Though we haven't got a chance to cross validate every model in our project, we relied on cross validation result to identify optimal parameters. It has significantly improved our efficiency in model testing and it's proven to be very reliable as to accuracy rate.

- Better understanding of R Packages

  Since R packages are open-sourced and written by so many different people, it's very important for us beginners to learn and understand the most common and reliable packages for each data mining technique. Being unfamiliar with all the functions and objects made our coding process very time-consuming. We wish we could have done more practices with these packages before start using them on large-scale data.

- Stay efficient with large dataset

  Last but not least, the number one issue we faced in our project is efficiency. With no experience in large-scale data, we found it extremely inefficient to try out models on the full dataset with traditional methodologies. Sampling data in support vector machine has helped us solve the efficiency issue and should be considered as a best practice in future data mining project.