Introduction

We will apply four statistical models of Naïve Bayes, Logistic Regression, Random Forest and Support Vector Machines on telecom data to predict whether or not a customer will leave Telco, a fictional telecom company. Predicting *customer churn* (also known as customer attrition) is important because across all industries, businesses are not only concerned with attracting new customers but also with retaining current customers. If a business can better understand what factors may cause a person to be more likely to leave, then the business can adjust its marketing and corporate strategy in hopes of lowering their customer churn. The telecommunications industry in particular is especially concerned with decreasing customer churn since, in most countries, a customer can choose from several different companies and the process of switching telecom providers is fairly easy. The ability to accurately predict which customers are more likely to churn can lead to significant monetary benefits for a company since, according to the *Harvard Business Review*, "acquiring a new customer is anywhere from 5 to 25 times more expensive than retaining an existing one," which means reducing customer churn is a significantly less costly option than increasing the number of new customers (Gallo).

Relevant Studies

Applying statistical analysis techniques to better predict customer attrition is not only a theoretical exercise, but is regularly conducted by various businesses across the globe. In a research study aimed to predict customer churn for SyriaTel customers with the same aim to create a model that identifies whether a person will churn, after going through the "features engineering phase," the authors applied the following analysis techniques on their data: Decision Tree, Random Forest, Gradient Boosting Machine (GBM) tree algorithm, and XGBOOST algorithm (Ahmad). Seventy percent of their sample data was allocated for training and the remaining thirty percent was allocated for testing. They measured the success of a model by its AUC value, and found the XGBOOST algorithm to have an AUC of 93.301%.

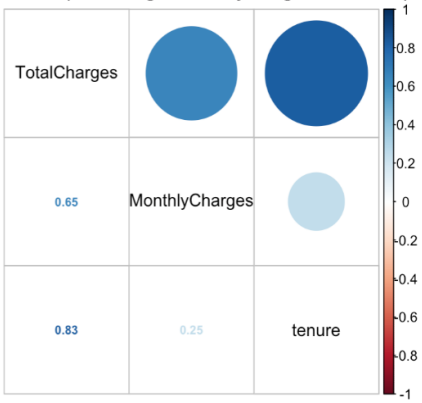
In a second study we looked at, the authors also analyzed the Telecom dataset. Two important assumptions made: each customer that left the company would cost $500 to replace and every customer could be retained by spending $100 on them. To measure their success, the authors compared their model's performance against a dummy model that spends $100 per customer. They used a stratified train-test-split to split their data into a training set and test set. In addition, in order to prevent their model from under-classifying their target variable, they used SMOTE from imblearn.over_sampling so that the minority class would be 50% of their dataset. After performing train-test-split, the authors plotted a ROC curve. They optimized the four models (Logistic Regression, Gradient Boosting, Random Forest, and AdaBoost) based on recall as their target metric. Through comparing the recall percentage and net amount of money saved amongst models, the authors determined that their Logistic Regression model performed best with a recall of 81% and net savings of $272,000 compared to the dummy model (Heintz).

Data Exploration

Our dataset contains customer information for a fictional telecom company, Telco. While we found our dataset on Kaggle, the dataset was originally put forth by IBM as the data module *Telco Customer Churn* (Telco). We are interested in predicting the churn value (either "Yes" or "No") for a particular customer. The churn value indicates whether a particular customer has left the company within the last month (resulting in a "Yes") or remains a customer at the company (resulting in a "No"). The raw data contains 7,043 rows with one row per customer and 21 features (explained in more detail below). Eleven of these rows had NA value for TotalCharges feature and hence were removed from the dataset.

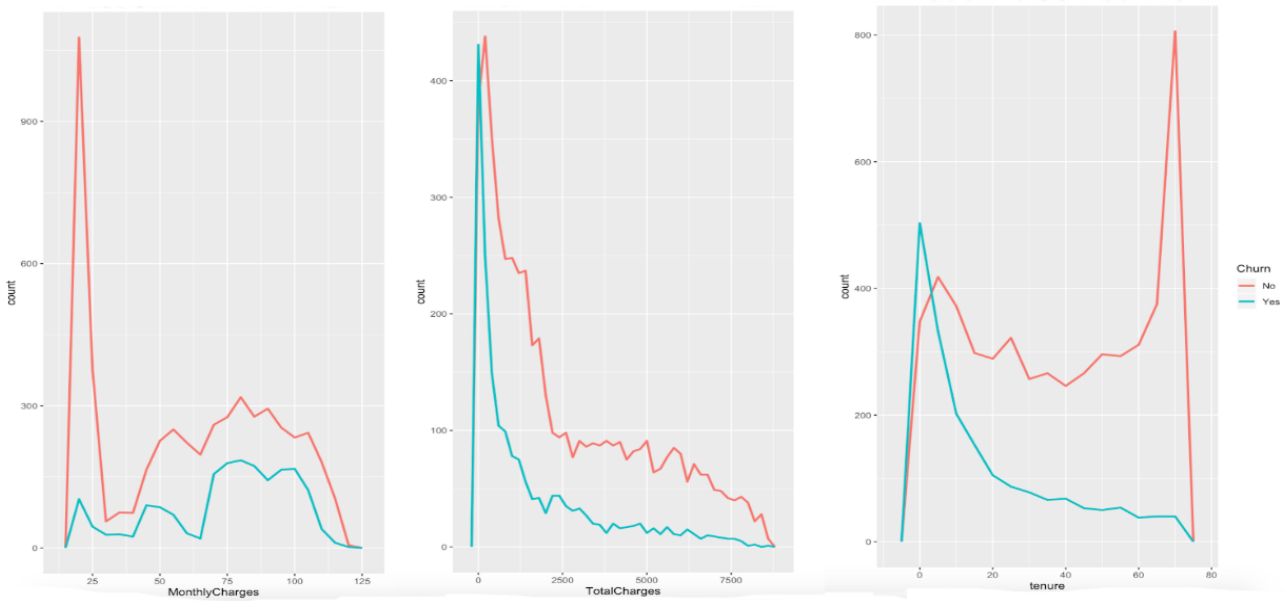
Numeric Features

Figure 1: Plot depicting correlations between numeric features (TotalCharges, MonthlyCharges, and tenure)



There are three numeric features of this dataset - Tenure (the number of months the person has been a customer), MonthlyCharges and TotalCharges. Multicollinearity, which occurs when two or more features are shown to be highly related, can lead to calculations involving an individual feature to be inaccurate, but it does not reduce predictive power and reliability of the model as a whole at least within the sample set. As seen in Figure 1, there appears to be a high correlation (0.85) between TotalCharges and tenure, a moderately high correlation between TotalCharges and Monthly charges (0.65), and a moderately low correlation between MonthlyCharges and tenure.

Figure 2: Relationship between Churn and Numeric Features



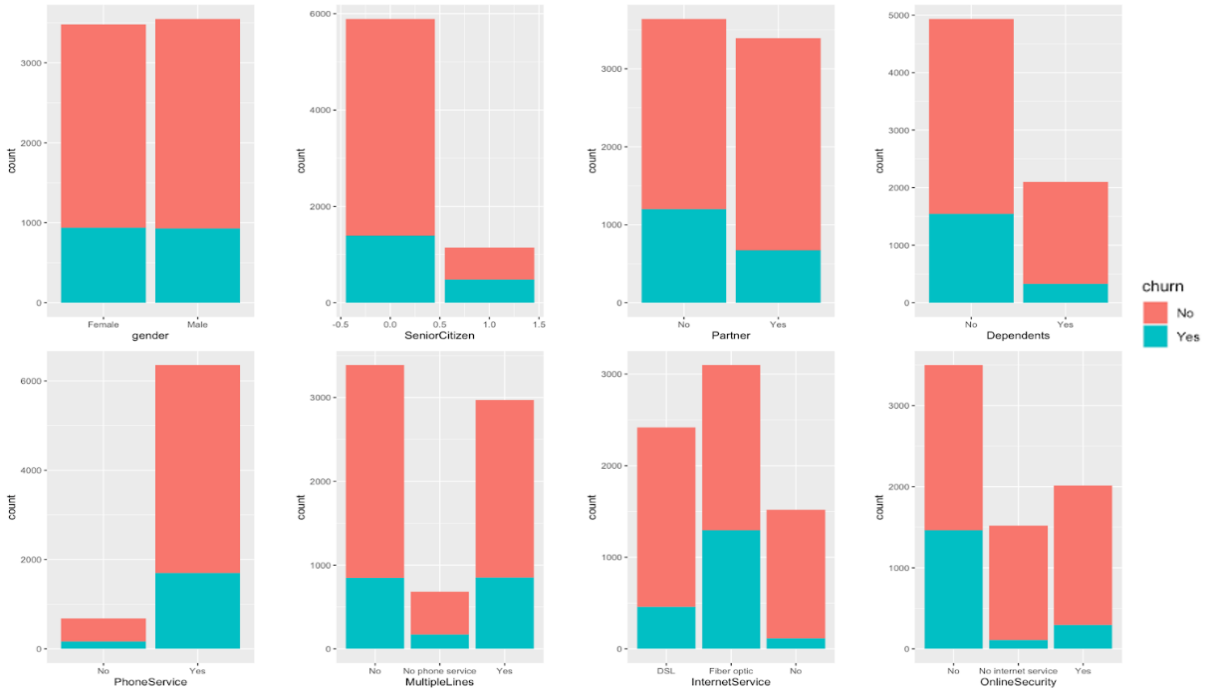
As observed in Figure 2, an extremely high number of customers have monthly charges

less than $25. The distributions of churn values for customers with monthly charges greater than $30 are quite similar. The distribution of total charges is positively skewed regardless if the customer churned or not. The distributions for tenure are markedly different between customers who churned and those who did not. Figure 2 shows that the distribution is positively skewed for customers who churned and negatively skewed for customers who have not churned. This leads us to conclude that customers who churn likely cancel the service in the first few months of joining the company. There appear to be two spikes, with the second spike denoting a large group of customers who have been using the service for more than five years and more prominent than the first spike.

Categorical Features

There are nineteen categorical features of this dataset - gender (male or female), Senior Citizen (1 for Yes, 0 for No), Partners (Yes or No), Dependants (Yes or No), PhoneService (Yes or No), MultipleLines (Yes, No and No phone Service), Contract (Month-to-Month, One Year, Two Year),PaperBiling (Yes or No), Payment Method (Bank Transfer, Credit Card, Electronic and Mail Check) and OnlineSecurity, OnlineBackUp, Device Protection, TechSupport, StreamingTV, StreamingMovies (all having Yes, No or No Internet Service as an input)

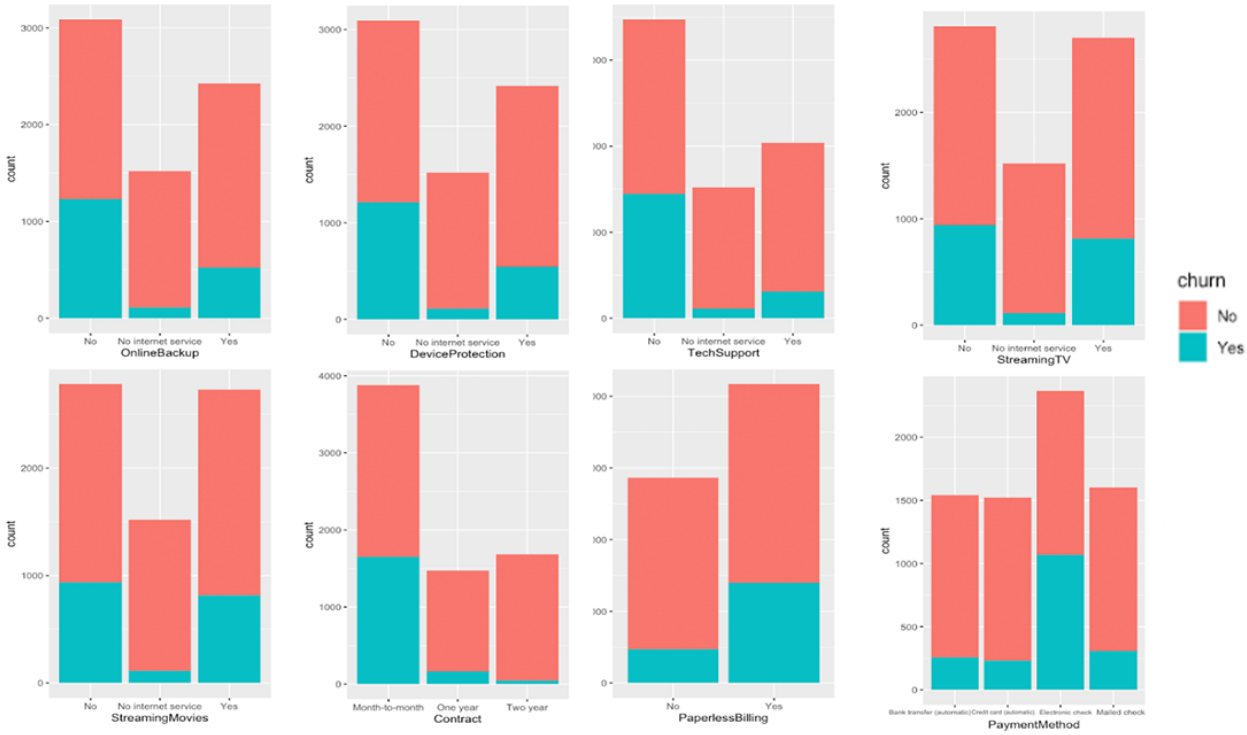
Figure 3: Relationship between Churn and from left to right (a) gender, (b) SeniorCitizen, (c) Partner, (d) Dependents, (e) PhoneService, (f) MultipleLines, (g) InternetService, (h) OnlineSecurity



If the value of PhoneService is "No," then the value of MultipleLines is "No phone service." This means that the "No phone service" column in Figure 3f does not actually hold any predictive power. Similarly, if the value of InternetService = "No," then the value of OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTv, and

StreamingMovies all had a value of "No internet service." Thus, the "No service column" for those features does not have any predictive power.

Figure 4: Relationship between Churn and from left to right (a) OnlineBackup, (b) DeviceProtection (c) TechSupport, (d) StreamingTV, (e) StreamingMovies, (f) Contract, (g) PaperlessBilling, (h) PaymentMethod



As seen in Figure 3h, Figure 4a, 4b, and 4c, customers who opted into these services (OnlineSecurity, OnlineBackup, OnlineBackup, and TechSupport, respectively) have a lower churn rate than customers who did not opt into these services. In contrast, Figure 3f, Figure 4d and 5e do not show a large difference in churn rates customers who do or do not have these services. Another observation we noticed is that an increase in contract length led to a decrease in churn rate as seen in Figure 4f. Additionally, the ways in which customers are billed and pay for Telco's services seem to have an impact on churn rates. in Figure 4g, customers with paperless billing have a higher churn rate than those without. In Figure 4h, customers who pay with electronic check have a higher churn rate than those who pay using alternative methods. However, one should not that for some features, the churn rate does not significantly differ. In Figure 3a and 3e, the churn rate remains fairly consistent for both response values. Demographics also seem to have some effect on churn rates. One may also note that senior citizens (value > 0.5) , as shown in Figure 3b, have a higher churn rate than those who are not senior citizens (value < 0.5). Customers with partners or dependents (Figure 3c and 3d, respectively) have a lower churn rate than customers who are single or do not have dependents.

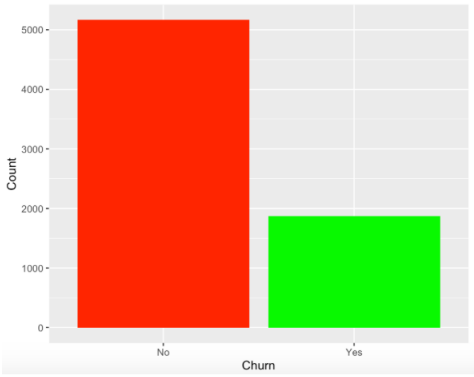
Methodology

Please note that if one wishes to replicate our work, R version 3.5.3 or below must be used in order to ensure the same result from set.seed(3354). Since the random generator in version 3.6.0 and onwards was modified, set.seed, which can be used to create reproducible random partitions of a dataset, will produce different results across the two versions (Bschneidr).

We performed the remainder of our statistical analysis on a dataset of 7,032 customer records. In addition, we removed the customerID column from all entries as a customer's ID was only present for unique identification and therefore would not be relevant to predicting customer churn. In order to determine which features to use to train our models, we used the Boruta R package. This package uses the Boruta algorithm to determine which features are uncorrelated and non-redundant.

The Boruta algorithm creates shuffled copies of all features, which are called shadow features, in order to add randomness to the dataset. A Random Forest classifier is trained on the dataset and a "feature importance measure" with a higher value corresponding to greater importance is applied. The Boruta algorithm does this by duplicating the dataset and then shuffling the values (called shadow features) in each column. Next, it trains a classifier on the dataset in order to establish the importance of each feature with higher scores corresponding to higher importance. The algorithm then compares the importance of the real features with the importance of the shadow features. If the feature has a higher Z-score, whereby the Z-score is the number of standard deviations a data point is from the mean, than the maximum Z-score of the shadow features, then it will be stored in a table, the feature will be marked as important, and the next iteration will occur (How to perform). The process of comparing the Z-score of the real feature with the Z-score of randomly-shuffled copies of the shadow feature removes features which are deemed highly unimportant (due to the better performance of the shadow feature). Finally, the algorithm stops either when all features are confirmed or rejected or it reaches a specified limit of Random Forest runs (How to perform).

Figure 5: Churn Value of Customer data (except NA)



After running the Boruta algorithm, we found the unimportant features to be Phone Service and Gender and removed these features when performing our statistical analysis. After removing certain features and data entries as detailed above, we used createDataPartition within the caret package to divide 80% of the data into a training set and the remaining 20% into a test set. Using createDataPartition allowed us to construct testing and training datasets while maintaining important groupings that existed within the complete dataset, which as evidenced in Figure 1, is relevant as only 1,869 of 7,043 customers (26.57%) had a churn value = "Yes" and we wanted to maintain this distribution in our training and test subsets.

After constructing our training and test datasets, in order to train our models, we used 10-fold cross validation. One of the reasons we decided to use cross validation was so that we

could train and test our models on all the data we had available, which is particularly useful since our dataset only contained 5,625 entries for training. In addition, performing cross validation gives us increased confidence in our model's performance. By training our model on multiple sets of data and using accuracy, i.e. the number of correct predictions divided by the total number of predictions, as our measurement of success, it will be easier to identify potential shortcomings in our algorithm since we can better predict how the algorithm would perform if given similar distributions of data (Shulga). In order to perform cross validation, we first ensured the dataset had been randomly shuffled. We then split the dataset into k = 10 folds, and for each iteration, we selected one fold (that had not been selected previously) to be the test dataset and used the remaining k-1 folds to train our model. We recorded the accuracy achieved on each iteration and after performing k iterations, set the model's performance score equal to the average accuracy of the k iterations (Sanjay.M). We decided to set k = 10 since 10 has been empirically shown to produce a test error rate estimate that does not have an extremely high bias or an excessively high variance (Gareth et al., 2015).

Naïve Bayes Algorithm

Naïve Bayes Algorithm is a widely used classification method in machine learning due to its simplicity. It is built on a classical theorem in probability, called Bayes' Theorem, which can be used to calculate the conditional probability of an event. Given a set of features X and a categorical response y (with n categories), Naïve Bayes Algorithm boils down a classification problem to finding the probability $P(y = y_k \mid X)$, for k= 1, 2, …, n using $P(X \mid y = y_k)$, $P(y = y_k)$ and P(X), with a strong assumption that the features are independent of one another (Ray).

In our classification problem of interest, we have to find, for a set of features of any given customer in the test set, P(Churn | features) using P(features | Churn) (i.e. likelihood of the observation given the customer has churned) and P(features) (i.e. unconditional probability of observing the features) with the formula given by Bayes' Theorem. If P(Churn | features) > 0.5 (or equivalently, P(Churn | features) > P(No Churn | features)), then the customer is predicted to churn.

<u>Assumption</u>

To calculate the likelihood of an observation conditioned on its churn indicator (or P(features | Churn)), mutual independence among features is assumed.

<u>Methodology</u>

A twist is worth paying attention to when performing classification with Naïve Bayes Algorithm in our data set – two continuous features are found in the data, namely MonthlyCharges and TotalCharges. While it is possible to compute the likelihoods by assuming that both conditionally follow a normal distribution, it is customary to apply Naïve Bayes

Algorithm to a data set with only categorical features. To compare with model performance with continuous features, they are recoded into four categories segmented by quartiles.

In short, we will compare the predictability of the following variations:

Variation 1	Continuous features are recorded as ***categorical*** features
Variation 2	Continuous features are preserved

Model performances on these variations are compared based on the cross-validation accuracy.

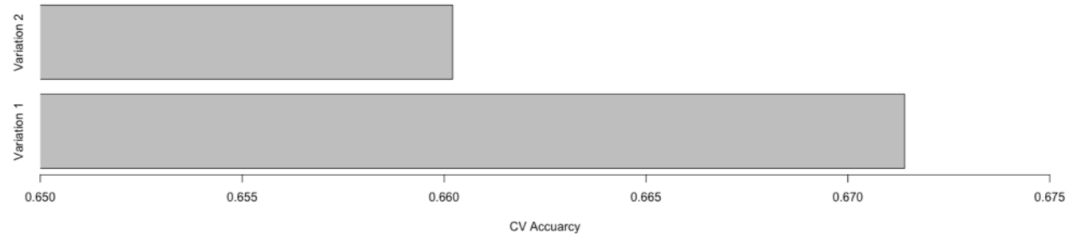
Model Comparison

Similar to previous models, two Naïve Bayes models will be trained and 10-fold cross validated on the training set with accuracy serving as the performance metric.

Variation	Description	Confusion Matrix*	CV accuracy	Test accuracy
1	Continuous features are recorded as ***categorical*** features	Reference Prediction No Yes No 43.824418 3.269948 Yes 29.589479 23.316154	0.6714	0.6641
2	Continuous features are preserved	Reference Prediction No Yes No 42.331615 2.896748 Yes 31.082282 23.689355	0.6602	0.6577

* Entries are shown as count percentages averaged over the 10 folds of cross-validation

Figure 6: CV Accuracy from Naive Bayes Algorithm



Discussion

With the highest CV accuracy on the training set as well as the test set, Variation 1 is considered the "best" variation of the Naïve Bayes model. This section explains why recoding the continuous features as categorical ones yielded a higher accuracy.

When the Naïve Bayes Algorithm is adopted with continuous features, the algorithm assumes they conditionally follow a normal distribution (Bouckaert). This is obviously not the case as seen in Figure 2 which shows skewed conditional distribution for numeric features. As Naïve Bayes Algorithm makes predictions by estimating conditional distributions of features with the training data, preserving these continuous features is not ideal as they seem to violate the normality assumption.

Furthermore, it is worth noting that Naïve Bayes Algorithm assumes there to be independence amongst features, which should be further verified against our data set.

Logistic Regression Algorithm

Another model we used to predict churn is Logistic Regression. In the Logistic Regression model, the dependent variable assumes two values: zero or one. Thus, this model is used for predictions in which there are only two possible outcomes, which makes the model suitable for our task of predicting whether a customer has churned. Logistic Regression uses the logistic function (with the features we are using):

$$p(X) = \frac{e^{\beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta nXn}}{1 + e^{\beta 0 + \beta 1X3 + \beta 2X2 + \ldots + \beta nXn}}$$

In the logistic function, p(X) is the probability of the particular data X and βj refers to the jth feature. We must estimate the coefficients βj for our j =1,2...n features. With this function, one should note that the result will be a number between 1 and 0; every time we obtain a feature greater than 0.5 we classify it as 1 and if it is lower we classify it as 0 (Gareth et al., 2015).

Methodology

We applied Logistic Regression to the Telco dataset by using the library caret to make the prediction, the GLM (generalized linear model) method, and setting the family to be binomial. We also performed 10-fold cross-validation k-fold at this stage.

First, we fit different Logistic Regressions to the training data set and evaluated the test data set; these new models are from the caret package. The models we used included BayesGLM (Bayesian Generalized Linear Model) and Logit Boost (referring to Boosted Logistic Regression). When applying these models, we only considered the features the Boruta Algorithm defined as important. BayesGLM uses the logistic function, however, it takes a different approach to obtain the coefficients βi. It employs some prior distributions for βi, which we assume will have a normal distribution. BayesGLM uses the output of the probability function as a parameter p of Bernoulli distribution that will describe Y, the prediction (Kruschke).

Boosted Logistic Regression differs from standard Logistic Regression because it creates a set of observations slightly different from each other. Predictions are then made for each one, and based on the predictions of all sets it will calculate a new prediction and will calculate a weight for each set depending on how correct its predictions were (Friedman). In this way, the

model has a tuning parameter nIter, which is equal to the number of boosting iterations. Since we ran the model without determining a particular nIter, R will run the model with nIter set to different numbers. It reported that the best accuracy was with nIter equal to 11. Therefore, we will have three different variations of the Logistic Regression.

Model Comparison

With all of the variations in mind, we ran the code and report the results of the 10-fold cross validation on the training to be as follows:

Variation	**Description**	**Confusion Matrix (in %)**	**CV accuracy**	**Test Accuracy**
1	Logistic Regression	Reference Prediction No Yes No 65.701084 11.782477 Yes 7.712813 14.803625	0.8058	0.7964
2	Bayes GLM	Reference Prediction No Yes No 65.683313 11.835792 Yes 7.730585 14.750311	0.8052	0.7957
3	Boosted Logistic	Reference Prediction No Yes No 64.421539 12.848765 Yes 8.992358 13.737338	0.7802	0.7779

Discussions

The highest CV accuracy was for the model using Logistic Regression (0.8058) so we will do further analysis for this model.

One potential explanation as to why the Logistic Regression provided a better accuracy than the Bayesian approach is because the data does not provide meaningful information; therefore, after updating the prior distribution we may have ended up with an uninformed posterior distribution. Regarding the boosted regression, one possible reason for its lower accuracy may be due to overfitting. The algorithm may have become too specific and may have been identifying the observations rather than its characteristics.

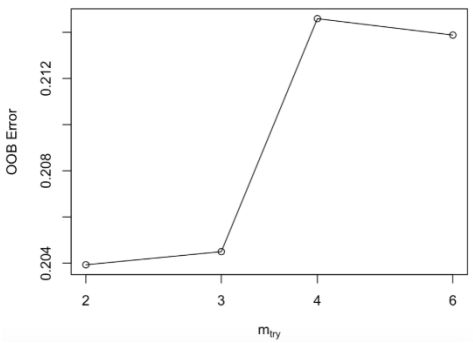
Random Forest Algorithm

The Random Forest algorithm works by aggregating the predictions made by choosing a random subset of features and building multiple decision trees of varying depth. Every decision tree in the forest is trained on a subset of the dataset called the bootstrapped dataset. The Random Forest is trained by using *bootstrap aggregation*, where each new tree is fit from a bootstrap sample of the training observations, and the portion of the dataset that was left out during the

construction of each decision tree in the forest is called the Out-Of-Bag (OOB) dataset (OOB Errors). The model uses the OOB dataset to automatically evaluate its own performance by running each of the samples in the OOB dataset through the forest. Out-of-bag error is an internal error estimate of the out-of-bag datasets. When used for classification and presented with a new dataset, Random Forest evaluates the final prediction by calculating the mode, or majority, of the predictions made by each individual decision tree in the forest (Maklin).

Methodology

Figure 8: RFtune using OOB Error to find best mtry



Random forest has some parameters that can be changed to improve the generalization of the prediction however, the 'rf' model contained in the Caret package only supports tuning of mtry, which is the number of features randomly sampled as candidates at each split. (By default, mtry is set equal to the square root of the number of features.) We used the tuneRF algorithm, provided in the RandomForest package. This algorithm searches the optimal mtry values given the data by starting with the default values and moving towards values of mtry with the lowest Out-of-Bag error estimate (Brownlee). As seen in Figure 8, the optimal value of mtry for our dataset is 2. By tuning our mtry to equal 2 and creating the 'rf' model, we obtained the plot shown below in Figure 9.

Model

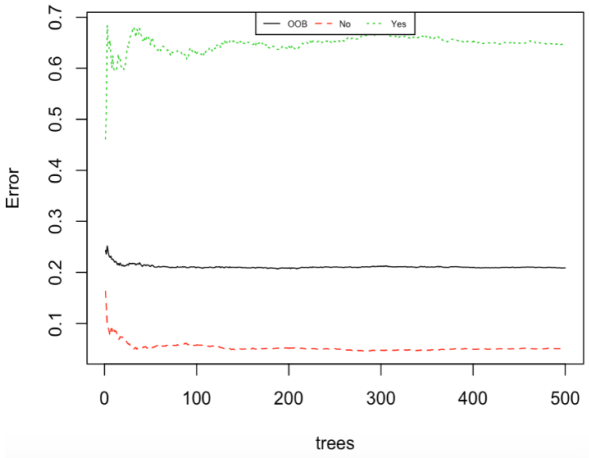
For our model, we have the following accuracies and CV confusion matrix :

		No	**Yes**
Prediction	**No**	69.65%	16.79%
	Yes	3.77%	9.79%

Type	**Accuracy**
Training (CV)	0.7944
Test	0.7794

Figure 9 shows the error rates on the training and test datasets as number of decision trees increases. The overall OOB error, the OOB error across the "No" class of Churn in the training dataset, and the OOB error across the "Yes" class of Churn in the training dataset are represented by the black, red, and green line, respectively.

Figure 9: Plot of Random Forest Model



The variable Importance shows the most significant attribute in decreasing order by mean decrease

in Gini. The mean decrease Gini can be viewed as a measure of the node purity at the end of the tree; the higher the mean decrease Gini, the better the homogeneity. This is because when a tree is built, the Gini impurity is calculated as part of the decision of which variable to split at (Hoare). For each variable, every time it is selected to split a node, the sum of the Gini decreases across every tree in the forest for that variable. The average Gini is obtained by dividing the total sum (across the number of trees) by the number of trees in the forest (Hoare). Based on the variable Importance, the four most important features for the Random Forest Model are tenure, TotalCharges, MonthlyCharges and InternetService.

Support Vector Machines

Support vector machines give us classifiers that maximize the margins between classes. They achieve this by identifying data points at the edge of the classes, and producing a separation hyperplane that maximizes the Euclidean distance (L2 norm) between those data points (also known as support vectors) and the hyperplane. Due to this maximization of distance, support vector machines generalize well. Although they work really well for truly separable cases, the large margin restriction becomes useless when the data has outliers that make it non-separable. To handle that, SVMs are augmented by two methods: kernels, and regularization.

Kernels are used to transform non-separable data into a separable form. They do so by mapping the data into a higher dimensional space in which the data is linearly separable. The most popular forms of kernels used are Polynomial, Gaussian, and linear (which is the same as a lack of a kernel). Kernels can allow the boundary between classes to curve, which can separate classes that are not linearly separable.

Regularization relaxes the separation boundary condition to allow misclassification for outliers. It gives a tradeoff between producing a separation boundary that correctly classifies all training data points and a boundary that has a large margin with the classes but allows some data to be misclassified. Regularization can improve test accuracy of an SVM model by reducing its variance. The regularization parameter (also known as cost) determines this tradeoff. A higher value of cost corresponds to “softer margins,” which means that the model gives more weight to increasing the margins than to ensuring that all of the data has been classified correctly.

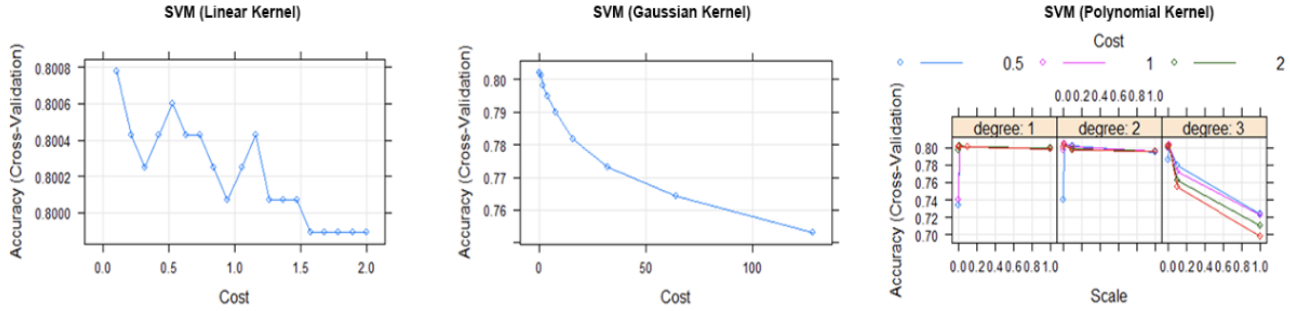
Methodology

We trained SVM models with linear, Gaussian, and polynomial kernels and compared their test accuracies. The tuning parameters for each kernel were determined using 10-fold cross validation. The Caret package was used for the implementation of the algorithms. The cost tuning parameter corresponds to regularization, sigma corresponds to the standard deviation parameter for the Gaussian kernel, degree corresponds to the highest power of the polynomial, and scale corresponds to the scaling of the polynomial.

Model Comparison

Kernel	Tuning parameters	CV accuracy	Test Accuracy	Confusion matrix (test)
Linear	Cost = 0.1052632	0.8008	0.7950	No Yes No 934 190 Yes 98 183
Gaussian	Sigma = 0.02645341 Cost = 0.25	0.8022	0.7943	No Yes No 960 217 Yes 72 156
Polynomial	Degree = 2 Scale = 0.01 Cost = 0.5	0.8045	0.7971	No Yes No 948 201 Yes 84 172

Figure 10: Plot of SVM Model



Discussions

Our results show that the polynomial kernel gives the best training error. Through cross validation, we get that a polynomial of degree 2 (quadratic kernel) is the most accurate model for our data which suggests that the natural separation boundary for the data is closest to a quadratic curve in shape. One reason why the polynomial kernel likely performs better is because the dataset can be linearly separated more easily in 3D space (Udiprod). In this case, linearly separated means that the data can separated for classification (Churn value of "Yes" or "No").

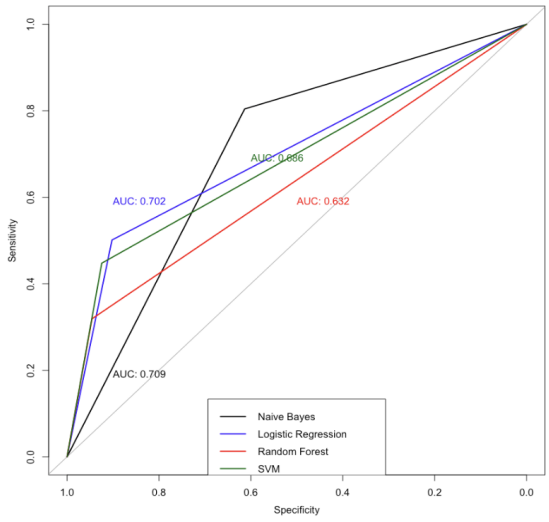
Comparison

In this section we compare all of the best sub-models for each method we tried. The best sub-models are the ones with the highest CV accuracy compared to the other sub-models for the same method. The training (CV) and test accuracy can be seen in the following table.

Accuracy	Naïve Bayes	Logistic Regression	Random Forest	SVM Polynomial
Train (CV)	0.6714	0.8058	0.7944	0.8045

Test	0.6641	0.7964	0.7794	0.7979

Figure 11: ROC Curve for all highest Train and Test Accuracy Models



The above table suggests that Logistic Regression as well as SVM Polynomial performed the best compared to the other models.

Another measure that we employed is the Receiver Operating Characteristics (ROC) curve for all of the models. The ROC curve is obtained by changing the threshold value and evaluating the combination of false positive rate and true positive rate for each threshold. While evaluating the models, we set the threshold as 0.5. This threshold can be interpreted as the value in which the model will decide if the Churn is equal to one or zero. If the probability returned by the model is higher than the threshold, the model assumes churn equals one.

One measure that we can use from the ROC curve is the area under the curve (AUC). While the ROC compares different thresholds, the AUC gives us an average of performance for various thresholds. It is possible to see that, although the Logistic Regression has an accuracy greater than the others for the threshold 0.5, when we consider the average performance for different thresholds, Naïve Bayes does a better job of predicting Churn status based on the ROC curve.

We find Logistic Regression to be the best model with its high prediction accuracy, which serves as a proxy for the actual performance of the classifier, as well as a high AUC value, which tells how well the two kinds of responses can be separated (Normanius). The simplicity of Logistic Regression is also worth noting. Regarding the Random Forest, when the dataset variance of the explanatory variables is high, the Logistic Regression tends to have a higher overall accuracy. Therefore, our features may have a significant variance, which may explain why the Logistic Regression performs better (Kirasich).

There is a similarity between the loss functions, which are used by algorithms to evaluate how well they model the data and improve themselves, of the Logistic Regression and support vector machine - Hinge Loss function to SVM and Logistic Loss to Logistic Regression. Therefore, they often produce similar results. It is possible to confirm this from the similarity that we observe in the accuracy of both models (Gareth et al., 2015).

It is worth mentioning that when the classes are separated we should prefer SVMs. However, if we have more overlapping regimes then we should prefer Logistic Regression as it should perform better as such is the case for our dataset (Gareth et al., 2015).

Conclusion

Customer retention is of great importance to telecom companies to determine effective business policies and remain competitive in the market. By analyzing the factors affecting customer churn, our project can help telecom companies understand their customers' needs, provide better services and generate greater profits in the long run.

After initial exploratory analysis, we used Boruta package to determine which of the features in the data were important enough to be included in our predictive models.We then trained different models of Naïve Bayes, Logistic Regression, Random Forest and SVM to see which model would give us the best cross-validation accuracy. It is important to note that each model has its pros and cons and that which model performs best depends on the intrinsic relationship between the response and features.

Our results show that Logistic Regression and SVM Polynomial gave the best test accuracy out of all the models tested. However, imbalanced responses on the data (~75% of Churn vs ~25% of No Churn in our data set) suggested that accuracy does not fully reflect a model's predictive power - if all observations are predicted to be No Churn, then an accuracy of 75% is already attained. Considering other evaluation metrics would be a solution to this. Comparing the ROCs of different models gave us a broader sense of model performance where, Naïve Bayes performed better (0.709 vs 0.702 of Logistic Regression). But with the high prediction accuracy (comparable to SVM Polynomial) as well as higher AUC, Logistic Regression is the best model for the dataset. To overcome the weakness of using accuracy as a performance metric in the imbalanced dataset, we could further look into F-score, which is the harmonic mean of true positive rate and precision, and might be a better indicator of the predictive ability of our models given the imbalance in target classes (Tharwat).

In fact, lack of positive churn responses has caused some of our models to produce a high number of false negatives as seen from the results. Besides using alternative evaluation metrics, another way to overcome this is re-sampling. Chawla, Bowyer, Hall, and Kegelmeyer (2002) proposed to adopt SMOTE, an over-sampling technique which generates new instances of the minority target class from feature values of existing ones, on the training data so as to balance Churn and No Churn responses.

Another limitation was the effect of the 'Internet Service' feature on the rest of the features. A value of 'No' for internet service would automatically mean that all online services would have a value of 'No internet service' which added an unnecessary third value to a binary variable which made our models more complex. The reason we did not choose to keep that feature in spite of its complications was that it accounted for approximately a fifth of the total data. In the future, given more data, we suggest removing the rows which had no internet service and then fitting the models again on the reduced data set to compare the results with those obtained in this study. It is expected that the predictive power of some models, e.g. Naïve Bayes, which holds a strong independence assumption, would improve after reducing the dependencies among features.

References

1. Ahmad, A.K., Jafar, A. & Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data 6*. https://doi.org/10.1186/s40537-019-0191-6
2. Bouckaert, Remco R. (2004). Naive bayes classifiers that perform well with continuous variables. *In Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence (AI 04),* 1089-1094. http://doi.org/10.1007/978-3-540-30549-1_106
3. Brownlee, Jason. (2016). True machine learning algorithms in R. Retrieved from https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/
4. Bschneidr (2019, May 30). Is set.seed consistent over different versions of R (and Ubuntu)? [Msg 2]. Message posted to https://stackoverflow.com/questions/47199415/is-set-seed-consistent-over-different-versions-of-r-and-ubuntu/56381613#56381613
5. Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer W. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research 16*, 321-357. https://doi.org/10.1613/jair.953
6. Friedman, J., Hastie, T. & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics, 28*(2), 337-407. https://doi.org/10.1214/aos/1016218223
7. Gallo, Amy. (2014). The Value of Keeping the Right Customers. *Harvard Business Review.* Retrieved from https://hbr.org/2014/10/the-value-of-keeping-the-right-customers
8. Gareth, J., Witten, D., Hastie, T., Tibshiranin, R. (2015). *An introduction to statistical learning: with Applications in R (6th ed)*. New York, NY: Springer.
9. Heintz, Brenner. (2018). Cutting the cord: Predicting customer churn for a telecom company. Retrieved from https://towardsdatascience.com/cutting-the-cord-predicting-customer-churn-for-a-telecom-company-268e65f177a5
10. Hoare, Jake. (n.d.). How is variable importance calculated for a random forest? [Web log post]. Retrieved from https://www.displayr.com/how-is-variable-importance-calculated-for-a-random-forest/
11. How to perform feature selection (i.e. pick important variables) using Boruta Package in R? (2016, March 22). [Web log post]. Retrieved from https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/
12. Kruschke, J. K., Aguinis, H., & Joo, H. (2012). The time has come: Bayesian methods for data analysis in the organizational sciences. *Organizational Research Methods.* https://doi.org/10.1177/1094428112457829
13. Maklin, Cory. (2019). *Random forest in R.* Retrieved from https://towardsdatascience.com/random-forest-in-r-f66adf80ec9
14. Normanius (2019, Sep 4). Reason of having high AUC and low accuracy in a balanced dataset. [Msg 2]. Message posted to https://stackoverflow.com/questions/38387913/reason-of-having-high-auc-and-low-accuracy-in-a-balanced-dataset
15. Kirasich, K.; Smith, T. & Sadler, B. (2018). Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets. SMU Data Science Review Vol. 1: No. 3, Article 9 Retrieved from https://scholar.smu.edu/cgi/viewcontent.cgi?article=1041&context=datasciencereview
16. *OOB Errors for Random Forests* (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html
17. Ray, Sunil. (2017). 6 easy steps to learn naive bayes algorithm with codes in Python and R. Retrieved from https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
18. Sanjay.M. (2018). Why and how to cross validate a model. Retrieved from https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f
19. Shulga, Dima. (2018). 5 reasons why you should use cross validation in your data science project. Retrieved from https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79
20. Telco Customer Churn. (2017). *Kaggle* [Data file]. Retrieved from https://www.kaggle.com/pavanraj159/telecom-customer-churn-prediction/data
21. Tharwat, Alaa. (2018). Classification assessment methods. *Applied Computing and Informatics.* https://doi.org/10.1016/j.aci.2018.08.003
22. Udiprod. (2007, Feb 5). *SVM with polynomial kernel visualization.* [Video file]. Retrieved from https://www.youtube.com/watch?v=3liCbRZPrZA