Telco Customer Churn Prediction

Sohum S.

April 5, 2020

Introduction

Telco is a technology service provider for phone, internet, and streaming services. The company's dataset provides information about customers, the services they have signed up for, and payment details. The goal of this project is to use the data to predict which customers are likely to churn, or leave the service. This analysis will help the company further understand why certain customers are leaving and potentially prevent customers from leaving. This dataset was obtained from kaggle (https://www.kaggle.com/blastchar/telco-customer-churn).

Analysis

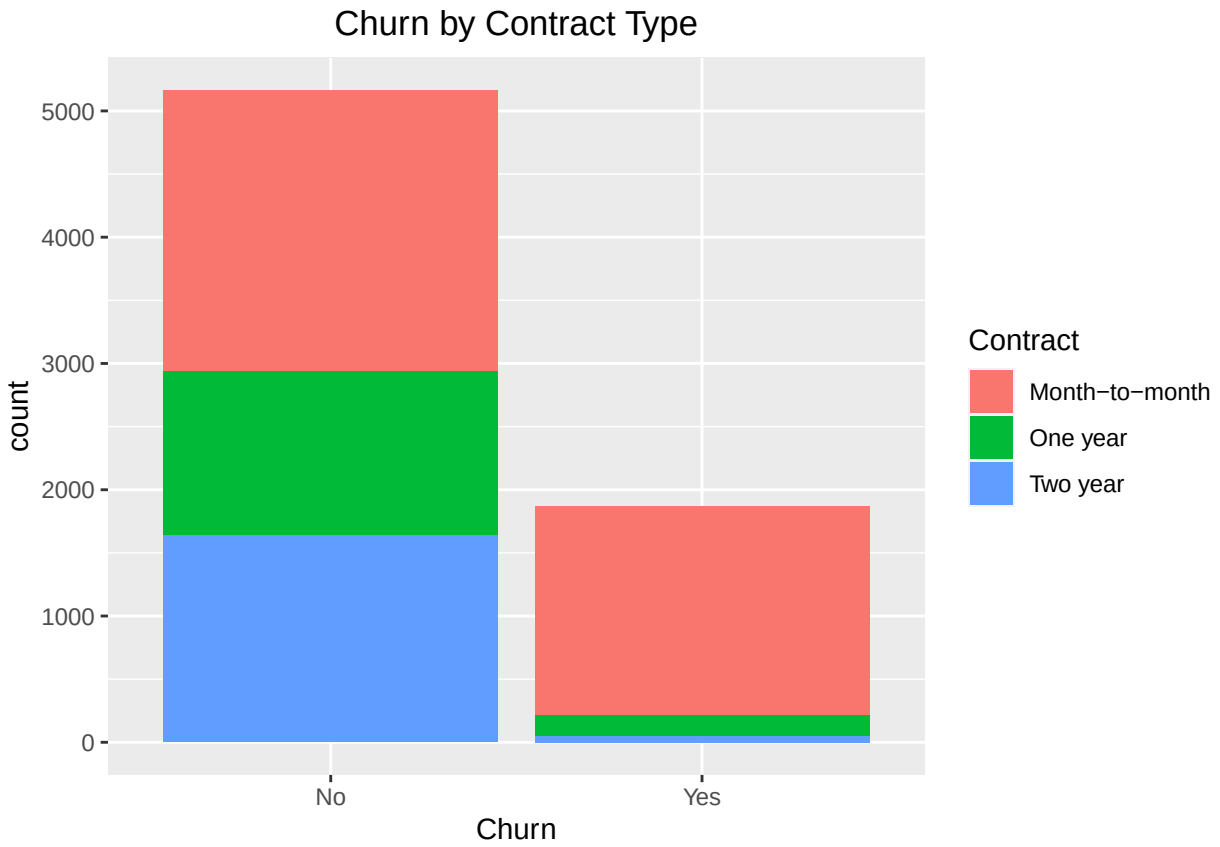
We begin by loading the dataset and removing any data with NA values.

```
#Load the dataset
cust_data = read.csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
#Remove rows with NA values
cust_data = na.omit(cust_data)
```

The "churn" variable is response variable which we aim to predict. We perform exploratory analysis using the predictor variables and identifying patterns that can be used for modeling.

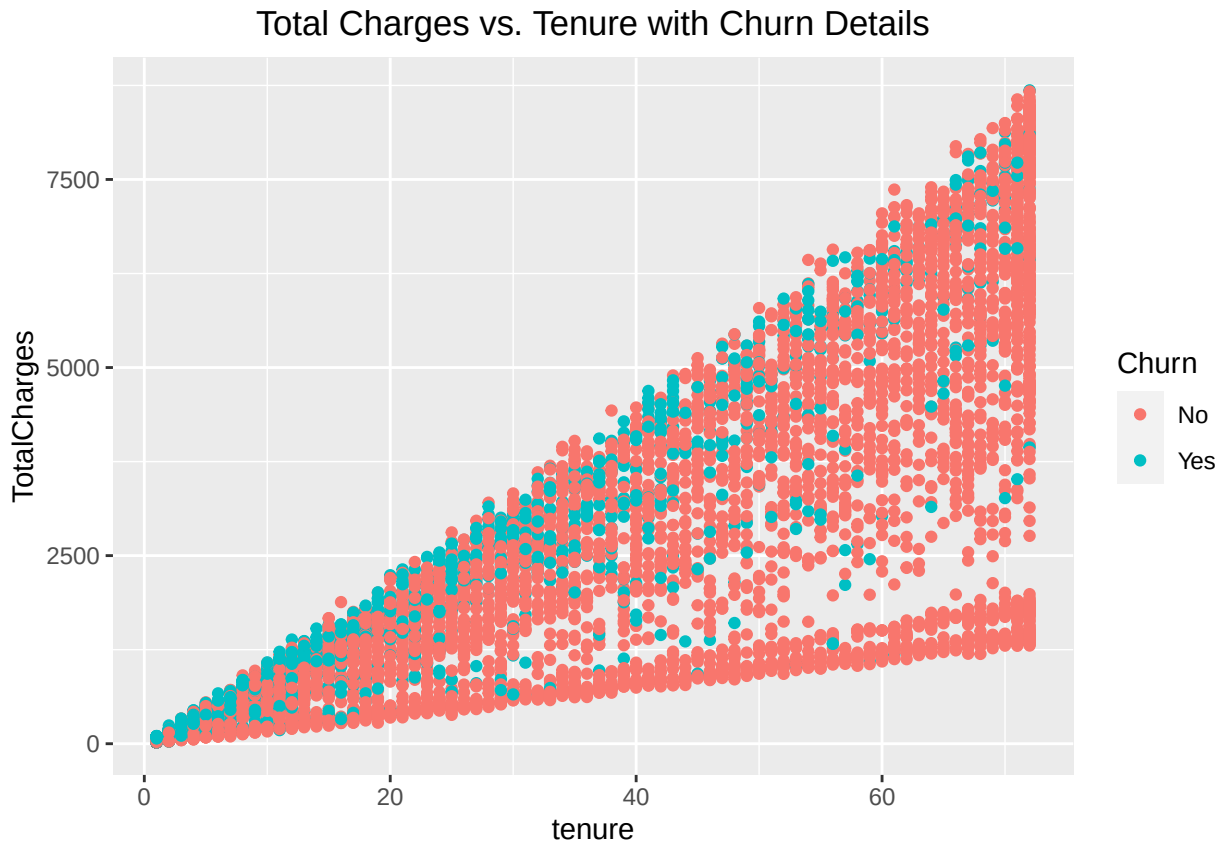
We start by making a stacked bar plot showing the churn based on the type of contract - month-to-month, one year, or two year. Based on the plot, customers that churned tend to have month-to-month contracts.

```
ggplot(cust_data, aes(fill = Contract, x = Churn)) +
  geom_bar(stat = "count") +
  ggtitle("Churn by Contract Type") +
  theme(plot.title = element_text(hjust = 0.5))
```



We also plot the tenure against total charges and use color to represent the churned/retained customers. Of course, the longer a customer stays, the more the higher their total charges will be. Based on the graph, customers who churned typically left after a short duration and paid higher costs than those who stayed.

```
ggplot(cust_data, aes(x = tenure, y = TotalCharges, color = Churn)) +
  geom_point() +
  ggtitle("Total Charges vs. Tenure with Churn Details") +
  theme(plot.title = element_text(hjust = 0.5))
```



We can also bucket customer tenure and monthly charge variables to analyze the proportion of customers in each bucket to leave their service.

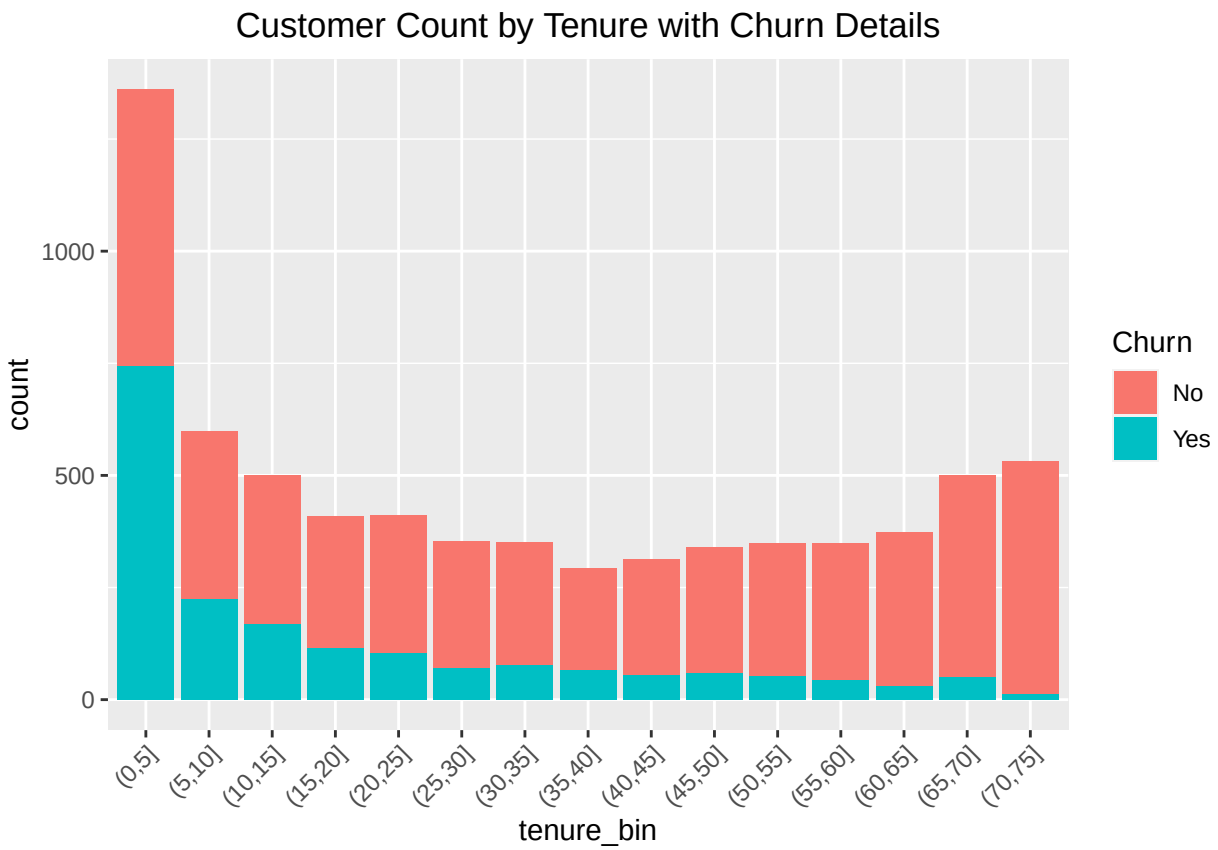
For tenure, we group customers' tenure in 5 month range buckets, ranging from 0 to 75 months. The table shows the distribution of customers who churned in each bracket. The bar plot shows a general trend that customers with a longer tenure tend to have lower churn rates. Most of the churn comes from customers who have not stayed with Telco for more than 20 months.

```
breaks_t = seq(0,75,5)
cust_data$tenure_bin = cut(cust_data$tenure, breaks_t)

tenure_bin_table = table(cust_data$tenure_bin, cust_data$Churn)
knitr::kable(tenure_bin_table)
```

	No	Yes
(0,5]	616	744
(5,10]	375	224
(10,15]	332	168
(15,20]	293	115
(20,25]	308	103
(25,30]	281	71
(30,35]	275	76
(35,40]	229	65
(40,45]	257	55
(45,50]	280	60
(50,55]	298	52

	No	Yes
(55,60]	305	43
(60,65]	344	30
(65,70]	450	51
(70,75]	520	12

```
ggplot(cust_data, aes(tenure_bin, fill = Churn)) +
  geom_bar() +
  ggtitle("Customer Count by Tenure with Churn Details") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```

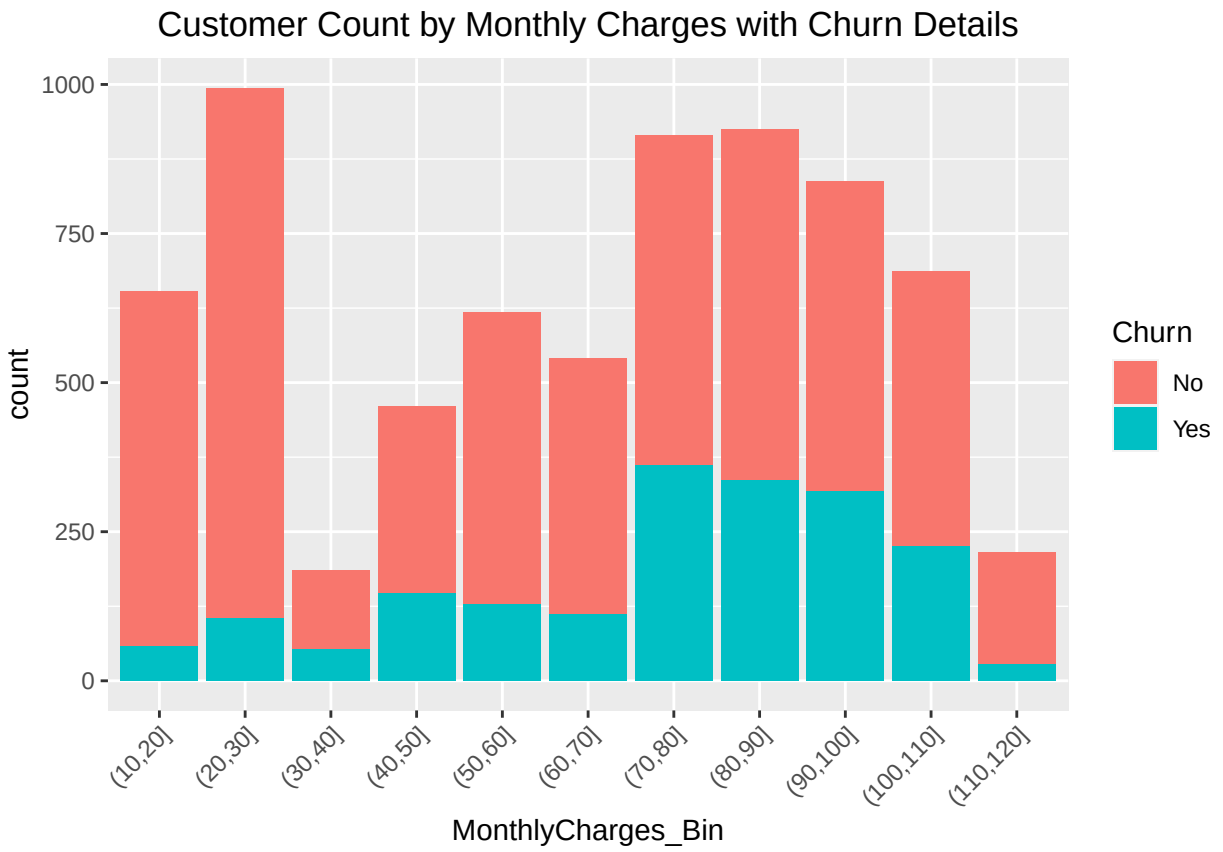


We group the customers' monthly charges into $10 range buckets, ranging from $10 to $120. The table shows the distribuion of customers who churned in each bracket, and the bar plot visualizes this information. There is a general trend that higher monthly rates tend to have greater probability of churn. It appears that customers in the $70-90 range tend to have a higher churn probability than other brackets.

```
breaks_mc = seq(10, 120, 10)
cust_data$MonthlyCharges_Bin = cut(cust_data$MonthlyCharges, breaks_mc)

mc_bin_table = table(cust_data$MonthlyCharges_Bin, cust_data$Churn)
knitr::kable(mc_bin_table)
```

	No	Yes
(10,20]	595	58
(20,30]	890	104
(30,40]	133	52
(40,50]	314	147
(50,60]	488	129
(60,70]	429	112
(70,80]	555	361
(80,90]	590	336
(90,100]	520	317
(100,110]	462	225
(110,120]	187	28

```
ggplot(cust_data, aes(MonthlyCharges_Bin, fill = Churn)) +
  geom_bar() +
  ggtitle("Customer Count by Monthly Charges with Churn Details") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```



From our analysis, we also should note that many variables are categorical. This means that when we begin modeling, we should convert these categorical variables into numerical variables, by encoding or using dummy variables.

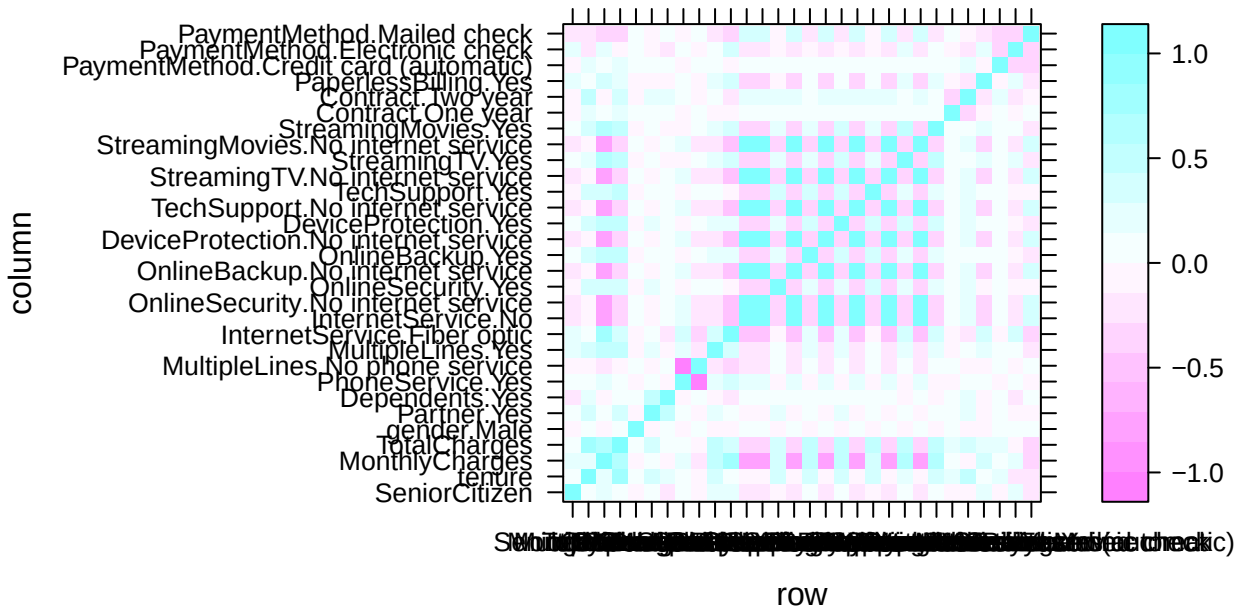
Prediction Methods

We can start by doing one-hot encoding for all categorical variables (except the response variable). We use the *dummyVars* function as part of the caret package to perform the encoding. Also, the general rule for dummy variables is to have one less variable than the number of categories, so we set the *fullRank* parameter to TRUE.

```
dv = dummyVars(~ .,
               data = cust_data[,c(2,4:5,7:18)],
               fullRank = TRUE)
enc_cust = predict(dv, newdata = cust_data)
#reconstruct the dataset - include the encoded variables now
cust_data_enc = cbind(cust_data[,-c(2,4:5,7:18)], enc_cust)

#remove columns not necessary for prediction
remove_cols = c("customerID", "tenure_bin", "MonthlyCharges_Bin")
cust_data_new = cust_data_enc[, -which(colnames(cust_data_enc) %in% remove_cols)]
```

We have to check which variables are correlated with one another before modeling. We plot a correlation matrix to show which variables are closely related to one another and remove the completely correlated variables.

```
#create correlation matrix
cust_cor = cor(cust_data_new[,-5])

#create correlation plot
cor_plot = levelplot(cust_cor)
cor_plot
```




```
#remove highly correlated variables
cust_data_mod = subset(cust_data_new,
                       select = -c(`OnlineSecurity.No internet service`,
                                   `OnlineBackup.No internet service`,
                                   `TechSupport.No internet service`,
                                   `StreamingTV.No internet service`,
                                   `StreamingMovies.No internet service`,
                                   `DeviceProtection.No internet service`,
                                   `MultipleLines.No phone service`))
```

From the plot and matching indices back to the variables, it is clear that the "No Internet Service" and "No Phone Service" dummy variables are completely correlated with the InternetService and PhoneService variables. We remove these variables and can begin our modeling efforts. There are possibly additional correlated variables, such as MonthlyCharges and Tenure, but we will analyze these after building initial models to get a better sense of which to remove.

Training and Test Sets

Using the cleaned customer dataset (cust_data_mod), we can create our training and test sets for modeling. We split the data so that 80% of the data is in the training set and 20% is in the test set. We will use only the training set for modeling and evaluate the performance on the test set.

```
#Build training and test sets
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = cust_data_mod$Churn, times = 1, p = 0.2, list = FALSE)

cust_train = cust_data_mod[-test_index,]
cust_test = cust_data_mod[test_index,]
```

Logistic Regression Model

We start modeling with logistic regression. We use all the predictor variables in our cust_train dataset to predict whether the customer is likely to churn or not. We then perform a VIF test to check for collinearity of the variables. Any variables with a VIF score greater than 5 have a chance of impacting the collinearity.

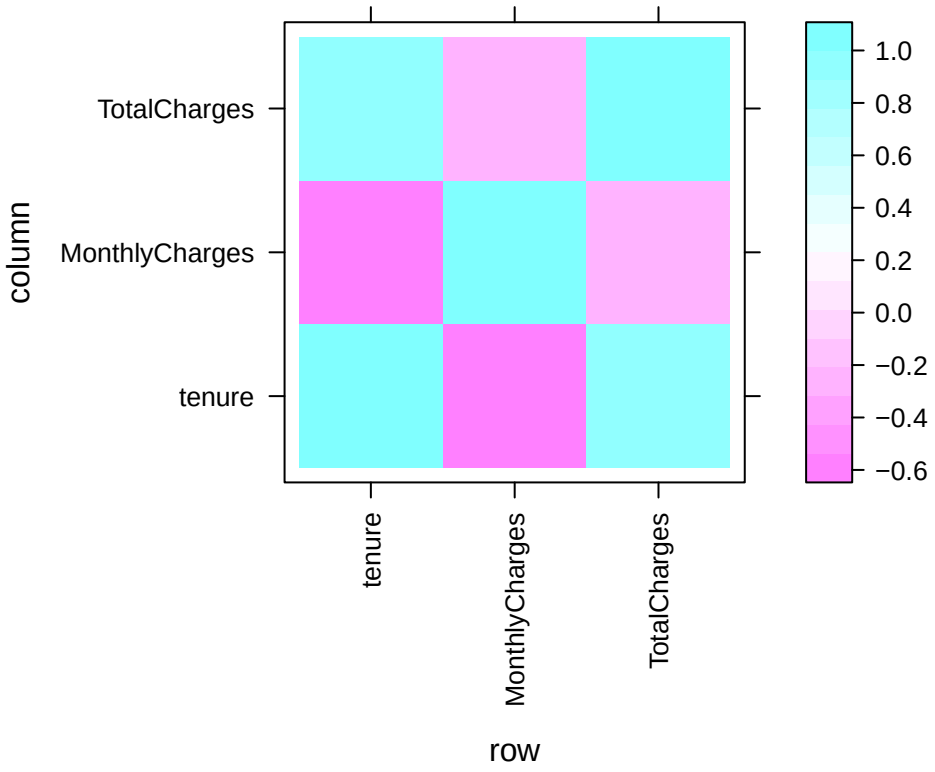
```
#Model 1 - use all variables
churn_log_model1 = glm(Churn ~ ., data = cust_train, family = binomial)
knitr::kable(coef(summary(churn_log_model1))[1:10,]) #show model coefficients summary for first 10 vari
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.4933987	0.9180799	1.6266544	0.1038105
SeniorCitizen	0.1979722	0.0943948	2.0972780	0.0359690
tenure	-0.0631081	0.0071324	-8.8480771	0.0000000
MonthlyCharges	-0.0512608	0.0358003	-1.4318564	0.1521849
TotalCharges	0.0003548	0.0000805	4.4092909	0.0000104
gender.Male	-0.0675043	0.0723665	-0.9328116	0.3509173
Partner.Yes	0.0018387	0.0864445	0.0212702	0.9830301
Dependents.Yes	-0.1834234	0.0994244	-1.8448538	0.0650588
PhoneService.Yes	0.4776095	0.7315060	0.6529126	0.5138126
MultipleLines.Yes	0.5032009	0.2003805	2.5112274	0.0120312

```
#Check for collinearity
vif_model1 = vif(churn_log_model1)
knitr::kable(vif_model1[vif_model1 > 5]) #show only collinear variables
```

	x
tenure	16.644752
MonthlyCharges	705.024213
TotalCharges	21.357851
PhoneService.Yes	33.462637
MultipleLines.Yes	7.541257
`InternetService.Fiber optic`	148.161114
InternetService.No	56.080095
OnlineSecurity.Yes	5.065585
OnlineBackup.Yes	6.518313
DeviceProtection.Yes	6.576935
TechSupport.Yes	5.254799
StreamingTV.Yes	25.204278
StreamingMovies.Yes	25.100715

```r
# Check correlation plot for tenure, TotalCharges, MonthlyCharges
cust_train_cor1 = cor(cust_train[2:4,2:4])
cust_train_corplot1 = levelplot(cust_train_cor1, aspect = "iso", scales=list(x=list(rot=90)))
cust_train_corplot1 #tenure, MonthlyCharges, and TotalCharges are highly correlated
```



Based on the results, it is clear there are some collinearity issues. It appears that tenure, MonthlyCharges, and TotalCharges are highly correlated with one another. MonthlyCharges has correlations with many other variables, so we should certainly remove it. Out of tenure and TotalCharges, we choose to retain tenure. The earlier exploratory analysis showed a clear trend that customers with shorter tenure tend to have a much higher churn rate.

```r
cust_train2 = subset(cust_train,
                     select = -c(MonthlyCharges, TotalCharges))

churn_log_model2 = glm(Churn ~ ., data = cust_train2, family = binomial)
#summary(churn_log_model2)
p_values_model2 = coef(summary(churn_log_model2))[,4]
knitr::kable(p_values_model2[p_values_model2 > 0.05])
```

	x
(Intercept)	0.2906651
gender.Male	0.3693936
Partner.Yes	0.9657786
OnlineBackup.Yes	0.1455875
DeviceProtection.Yes	0.5979038

	x
PaymentMethod.Credit card (automatic)	0.4073461
PaymentMethod.Mailed check	0.9309939

```
vif_model2 = vif(churn_log_model2)
vif_model2[vif_model2 > 5] #there is no collinearity
```

```
## named numeric(0)
```

In our second iteration of the model, we can see there is no more collinearity. Now, we can just remove the variables with low predictive power (P-value < 0.05). These variables are the gender, partner status, online backup (y/n), and device protection (y/n).

```
#Model 3 - remove variables that are not statistically significant
cust_train3 = subset(cust_train2,
                     select = -c(gender.Male, Partner.Yes,
                                 OnlineBackup.Yes, DeviceProtection.Yes))

churn_log_model3 = glm(Churn ~ ., data = cust_train3, family = binomial)
knitr::kable(coef(summary(churn_log_model3)))
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.2257883	0.1656874	-1.3627371	0.1729654
SeniorCitizen	0.1950909	0.0940975	2.0732849	0.0381458
tenure	-0.0353104	0.0025025	-14.1103001	0.0000000
Dependents.Yes	-0.1987089	0.0906511	-2.1920192	0.0283781
PhoneService.Yes	-0.3896579	0.1440492	-2.7050326	0.0068298
MultipleLines.Yes	0.3006218	0.0875524	3.4336204	0.0005956
InternetService.Fiber optic	0.8416984	0.1016099	8.2836279	0.0000000
InternetService.No	-0.8464335	0.1505103	-5.6237562	0.0000000
OnlineSecurity.Yes	-0.4088304	0.0939865	-4.3498844	0.0000136
TechSupport.Yes	-0.3721129	0.0953512	-3.9025498	0.0000952
StreamingTV.Yes	0.2926183	0.0885085	3.3061041	0.0009460
StreamingMovies.Yes	0.2683375	0.0884653	3.0332520	0.0024193
Contract.One year	-0.6451145	0.1186124	-5.4388439	0.0000001
Contract.Two year	-1.3270481	0.1932294	-6.8677355	0.0000000
PaperlessBilling.Yes	0.3242691	0.0827947	3.9165433	0.0000898
PaymentMethod.Credit card (automatic)	-0.1060621	0.1266317	-0.8375636	0.4022758
PaymentMethod.Electronic check	0.2809328	0.1045861	2.6861376	0.0072283
PaymentMethod.Mailed check	-0.0128006	0.1262173	-0.1014170	0.9192194

This is the final logistic regression model we can use for making predictions. We can attribute the predictive power to the selected variables and describe this power using odds ratios. The closer this odds ratio is lower to 0, the lesser the risk of customer churn. For example, the Contract.TwoYear variable has an odds ratio of $e^{-}1.327 = 0.2653$. It makes sense that customers with longer contracts are less likely to churn compared to those with shorter contracts. This means that of those customers with an odds ratio of 1/4, only 1 in 5 customers will churn. The PaperlessBilling variable has an odds ratio of $e^{0}.3243 = 1.383$, so those with paperless billing are more likely to churn compared to those without paperless billing.

Now, we have a model we can apply to make predictions. We need to adjust the test set to remove the

variables done throughout the model selection process, and then apply the logistic regression equation given by the final model. We denote the final datasets as cust_data_train and cust_data_test. These datasets can be effectively used for any further modeling approaches, as we are finished with all data cleaning, encoding, and variable selection.

```
#Setup Train and Test Sets
cust_train_f = cust_train3
cust_test_f = cust_test[,colnames(cust_train_f)]
```

For this section, we will only evaluate the model results on the training set. Later in the results section, we will apply our model on the test set. We start by applying the model equations to get predicted probabilities. Then, we construct a function to find the optimal cutoff for classification. Finally, we show the overall accuracy of the model on the training set. We will get into all the model metrics in the Results section, when applying the techniques on the test set.

```
#Predictions on Training Set
glm_churn_probs_train = predict(churn_log_model3, data = cust_train_f,
                                type = "response")
#Find the optimal cutoff level for deciding Yes/No
cutoffs = seq(0.2, 0.8, 0.01)

optimal_cutoff_func = function(i) {
  glm_churn_pred_train = rep("No", length(glm_churn_probs_train))
  glm_churn_pred_train[glm_churn_probs_train > i] = "Yes"
  accuracy = mean(glm_churn_pred_train ==  cust_train_f$Churn)
  return(accuracy)
}

cutoff_accs = sapply(cutoffs, optimal_cutoff_func)
opt_cutoff = cutoffs[which.max(cutoff_accs)] #cutoff of 0.55

glm_churn_pred_train = rep("No", length(glm_churn_probs_train))
glm_churn_pred_train[glm_churn_probs_train > opt_cutoff] = "Yes"

table(glm_churn_pred_train, cust_train_f$Churn)
```

```
##
## glm_churn_pred_train   No  Yes
##                   No  3769  742
##                   Yes  361  753
```

```
mean(glm_churn_pred_train ==  cust_train_f$Churn) #80.39111% accuracy
```

```
## [1] 0.8039111
```

We find the best cutoff for classifying is 0.55, so any probability above this value will be considered a churn. Our model has 80.39% accuracy on the training set. The baseline accuracy if we were to guess all customers to stay is roughly 73.42%, so the model certainly is valuable at making predictions. As we saw earlier, this model can easily be interpreted as well, making it easy to understand how each variable impacts the chance of customer churn.

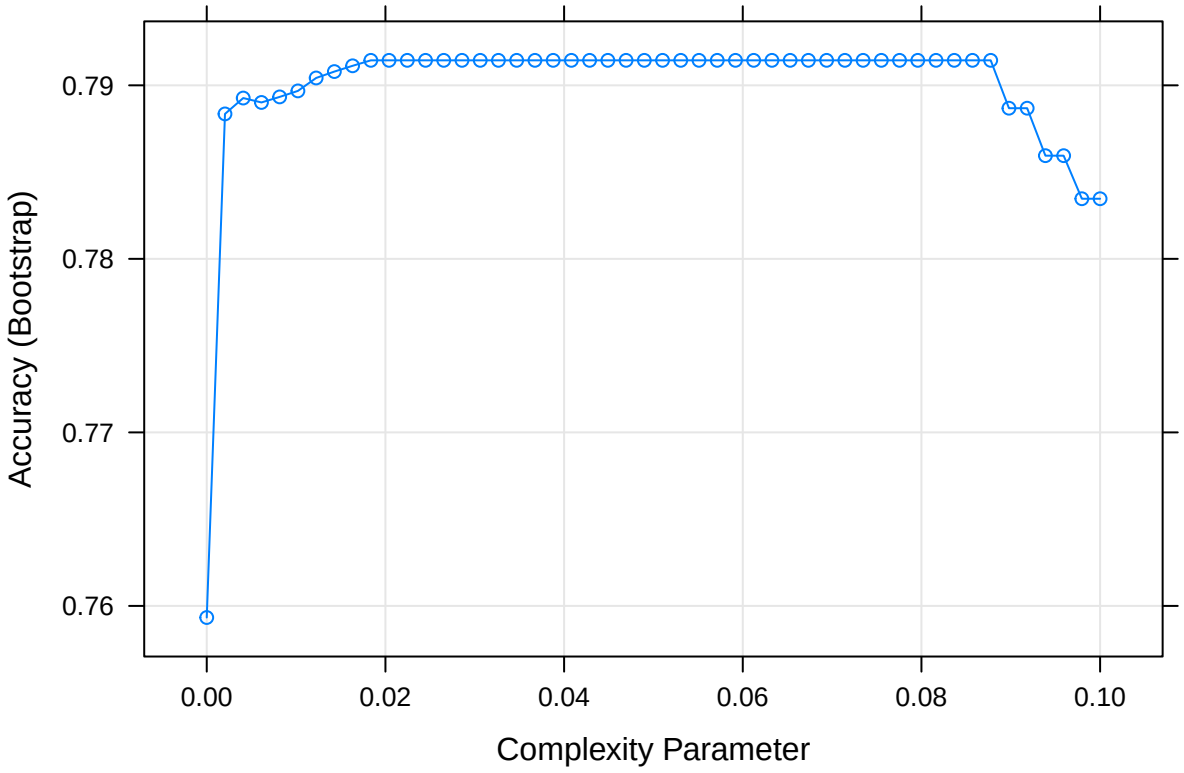
Decision Tree

Now we try building a decision tree model, using the train function as part of the caret package. We need to rename the columns to remove spaces to prevent the function from throwing errors. The results on the training set are shown below. Plots descrbing the parameter tuning and decision splitting are also included.

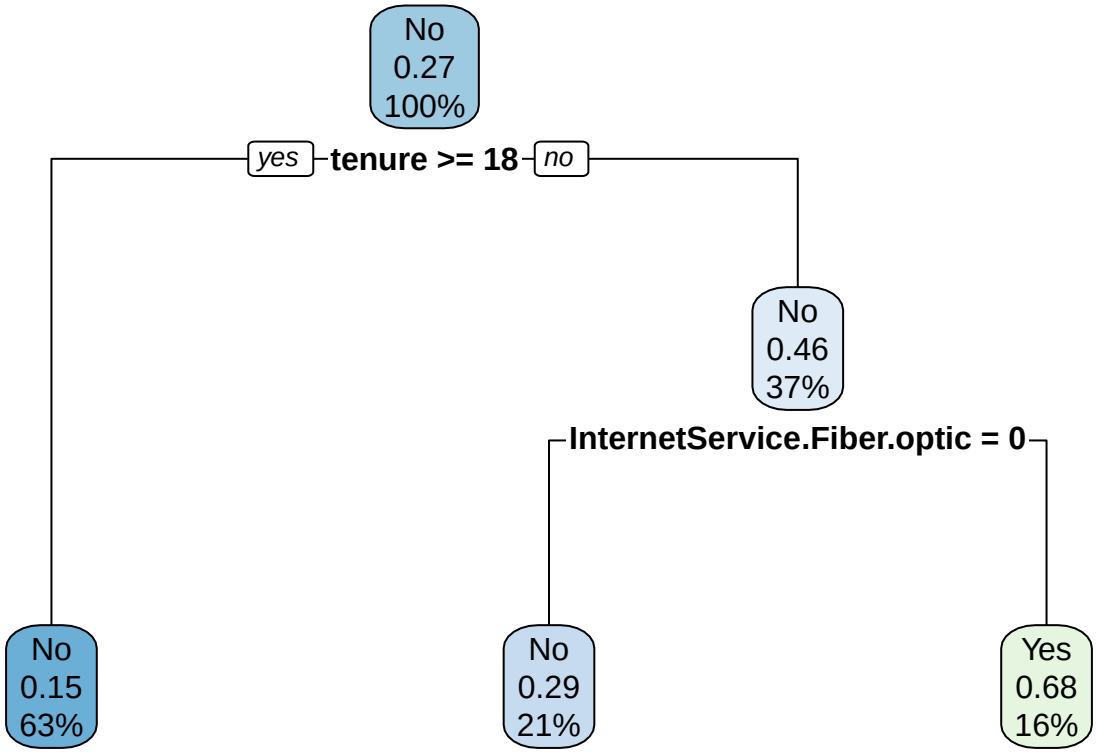
```
#Format column names (remove spaces)
colnames(cust_train_f) <- make.names(colnames(cust_train_f))
colnames(cust_test_f) <- make.names(colnames(cust_test_f))

#Build Decision Tree on Training Set
churn_train_rpart <- train(Churn ~ ., data = cust_train_f,
                           tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 50)),
                           method = "rpart")
confusionMatrix(churn_train_rpart)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  68.7 15.9
##        Yes  5.0 10.4
##
##  Accuracy (average) : 0.7914
```

```
#Plots
plot(churn_train_rpart)
```




```
rpart.plot(churn_train_rpart$finalModel)
```



The plot shows how tenure and internet service type are important variables in deciding customer churn.

Random Forest

We also build a random forest model to predict churn. The results on the training set are displayed below.

```
churn_train_randForest <- randomForest(Churn ~ ., data = cust_train_f,
                                       ntrees = 1000)
churn_train_randForest$confusion
```

```
##       No Yes class.error
## No  3697 433   0.1048426
## Yes  749 746   0.5010033
```

The performance on the training set is similar to logistic regression performance, so advanced methods might not be completely useful.

Results

We apply our 3 models (logistic, decision tree, and random forest) on the test set and summarize the results.

```
#Logistic Regression
cust_train_f = cust_train3
cust_test_f = cust_test[,colnames(cust_train_f)]
```

```
glm_churn_probs_test = predict(churn_log_model3, cust_test_f,
                               type = "response")
glm_churn_pred_test = rep("No", length(glm_churn_probs_test))
glm_churn_pred_test[glm_churn_probs_test > opt_cutoff] = "Yes"

glm_tbl = table(glm_churn_pred_test, cust_test_f$Churn)

glm_metrics = c(mean(glm_churn_pred_test ==  cust_test_f$Churn), #81.59204% accuracy
recall(glm_tbl), #91.48% recall
precision(glm_tbl)) #84.68% precision

#Decision Trees
#Format column names (remove spaces)
colnames(cust_train_f) <- make.names(colnames(cust_train_f))
colnames(cust_test_f) <- make.names(colnames(cust_test_f))

pred_churn_test_rpart = predict(churn_train_rpart, cust_test_f)
rpart_tbl = table(pred_churn_test_rpart, cust_test_f$Churn)

rpart_metrics = c(mean(pred_churn_test_rpart ==  cust_test_f$Churn), #78.32267% accuracy
recall(rpart_tbl), #88.577% recall
precision(rpart_tbl)) #83.03% precision

#Random Forest
pred_churn_test_rf = predict(churn_train_randForest, cust_test_f)
rf_tbl = table(pred_churn_test_rf, cust_test_f$Churn)

rf_metrics = c(mean(pred_churn_test_rf ==  cust_test_f$Churn), #80.31% accuracy
recall(rf_tbl), #90.13% recall
precision(rf_tbl)) #84.18% precision

#Result Summary
full_metrics = data.frame(glm_metrics, rpart_metrics, rf_metrics)
rownames(full_metrics) = c("accuracy", "recall", "precision")
knitr::kable(full_metrics)
```

	glm_metrics	rpart_metrics	rf_metrics
accuracy	0.8159204	0.7818053	0.8002843
recall	0.9148112	0.9254598	0.8935140
precision	0.8467742	0.8060708	0.8436929

Based on our results, it appears the logistic regression model is superior to the others. The accuracy of prediction is the highest. The precision, which is the fraction of actual churned out of those predicted to churn, is 91.48%. The recall, which is the fraction of true positives out of total actual positives, is 84.68%. The recall is the more important metric here, since we want to ensure that there are minimal false negatives. It is expensive for the company if they are not able to capture customers that are likely to churn, and a higher recall will minimize these false negative cases. The random forest model also performs relatively well, but due to the logistic model's simplicity and interpretive power, it is better to use the logistic model for predictions.

Conclusion

By using various approaches, we attempted to predict customer churn. We start by performing exploratory data analysis to understand trends in the data. We focus heavily on logistic regression and try to remove insignifiant and collinear variables. The three models tried include logistic regression, decision trees, and random forest. We conclude the logistic regression model performs the best on the bases of accuracy, recall, and precision. Further modeling approaches can also be tested, but given the logistic model already performs well, it is probably better to continue modifying the logistic equation. Aside from the models used, some cross validation procedures can be tried to verify the results. Some additional metrics, such as AUC/ROC and F1 score can be calculated to get a better understanding of performance.

Based on this model, we are able to interpret the regression coefficients as odds ratios and understand how each variable directly influences the probability of a churn. While all the selected variables are significant, the most significant ones were tenure, internet service choice, and contract duration. Customers with longer tenure were less likely to churn. Customers with fiber optic service compared to those without were more likely to churn, which might indicate problems with this particular service. Customers with longer contracts were less likely to churn. Ultimately, the logistic model provides useful information to help Telco in better understanding why their customers are leaving, and hopefully prevent further churn.

References

Dataset: https://www.kaggle.com/blastchar/telco-customer-churn

https://rafalab.github.io/dsbook/

http://faculty.marshall.usc.edu/gareth-james/ISL/

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

http://appliedpredictivemodeling.com/