## Overview

### Background

The telecommunications industry is facing a challenging future, with the emergence of new technologies and the increasing levels of competition resulting in unprecedented levels of customer attrition and price competition between the companies in the sector.

In a highly price sensitive market, customer service and product features play an important role. For telecommunication companies to better tailor their products to customer expectations, understanding the reasons for customer attrition is a crucial step. In that sense, customer churn analysis is one of the vital measures for subscription-based business models such as telecom services and internet providers.

In this example, we will explore multiple models to address the following question: Why a specific telecommunications company's customers churn? In this analysis, we are particularly interested in understanding differences between groups of customers, and to leverage that information to suggest future customer segmentation strategies.

We will also use modelling to predict which customers are likely to churn in the future.

We argue that if a company is able to predict if a customer is likely to churn, while also being able to identify if the same customer is worth retaining (based on a predicted value for CLTV), then the company can choose to increase engagement with the customer in order to retain the customer.

Preserving "at risk" valuable customers, while leveraging on information about differences in groups to develop better segmentation strategies, can potentially help a telecommunication company differentiate from the competition and hence increase revenues in the long run.

### Data Description

**The response variable is:**

*Churn Value:* 1 = the customer left the company. 0 = the customer remained with the company.

**The predicting variables are:**

*Country:* The country of the customer's primary residence.

*State:* The state of the customer's primary residence.

*City:* The city of the customer's primary residence.

*Zip Code:* The zip code of the customer's primary residence.

*Lat Long:* The combined latitude and longitude of the customer's primary residence.

*Latitude:* The latitude of the customer's primary residence.

*Longitude:* The longitude of the customer's primary residence.

*Gender:* The customer's gender: Male, Female

*Senior Citizen:* Indicates if the customer is 65 or older: Yes, No

*Partner:* Indicates if the customer is married: Yes, No

*Dependents:* Indicates if the customer lives with any dependents: Yes, No. Dependents could be children, parents, grandparents, etc.

*Tenure in Months:* Indicates the total amount of months that the customer has been with the company by the end of the quarter specified above.

*Phone Service:* Indicates if the customer subscribes to home phone service with the company: Yes, No

*Multiple Lines:* Indicates if the customer subscribes to multiple telephone lines with the company: Yes, No

*Internet Service:* Indicates if the customer subscribes to Internet service with the company: No, DSL, Fiber Optic, Cable.

*Online Security:* Indicates if the customer subscribes to an additional online security service provided by the company: Yes, No

*Online Backup:* Indicates if the customer subscribes to an additional online backup service provided by the company: Yes, No

*Device Protection:* Indicates if the customer subscribes to an additional device protection plan for their Internet equipment provided by the company: Yes, No

*Tech Support:* Indicates if the customer subscribes to an additional technical support plan from the company with reduced wait times: Yes, No

*Streaming TV:* Indicates if the customer uses their Internet service to stream television programing from a third party provider: Yes, No. The company does not charge an additional fee for this service.

*Streaming Movies:* Indicates if the customer uses their Internet service to stream movies from a third party provider: Yes, No. The company does not charge an additional fee for this service.

*Contract:* Indicates the customer's current contract type: Month-to-Month, One Year, Two Year.

*Paperless Billing:* Indicates if the customer has chosen paperless billing: Yes, No

*Payment Method:* Indicates how the customer pays their bill: Bank Withdrawal, Credit Card, Mailed Check

*Monthly Charges:* Indicates the customer's current total monthly charge for all their services from the company.

*Total Charges:* Indicates the customer's total charges, calculated to the end of the quarter specified above.

*Churn Label:* Yes = the customer left the company this quarter. No = the customer remained with the company. Directly related to Churn Value.

*CLTV:* Customer Lifetime Value. The higher the value, the more valuable the customer.

*Churn Score:* A value from 0-100 that is calculated using the predictive tool IBM SPSS Modeler. The model incorporates multiple factors known to cause churn. The higher the score, the more likely the customer will churn.

*Churn Reason:* A customer's specific reason for leaving the company. Directly related to Churn Category.

## Preparing the Data

### Reading data

```
# Set color codes
gtblue = rgb(0, 48, 87, maxColorValue = 255)
techgold = rgb(179, 163, 105, maxColorValue = 255)
buzzgold = rgb(234, 170, 0, maxColorValue = 255)
bobbyjones = rgb(55, 113, 23, maxColorValue = 255)

# Read the data using read_csv
dat <- read.csv("Telco_customer_churn.csv", fileEncoding="UTF-8-BOM", header=TRUE)
```

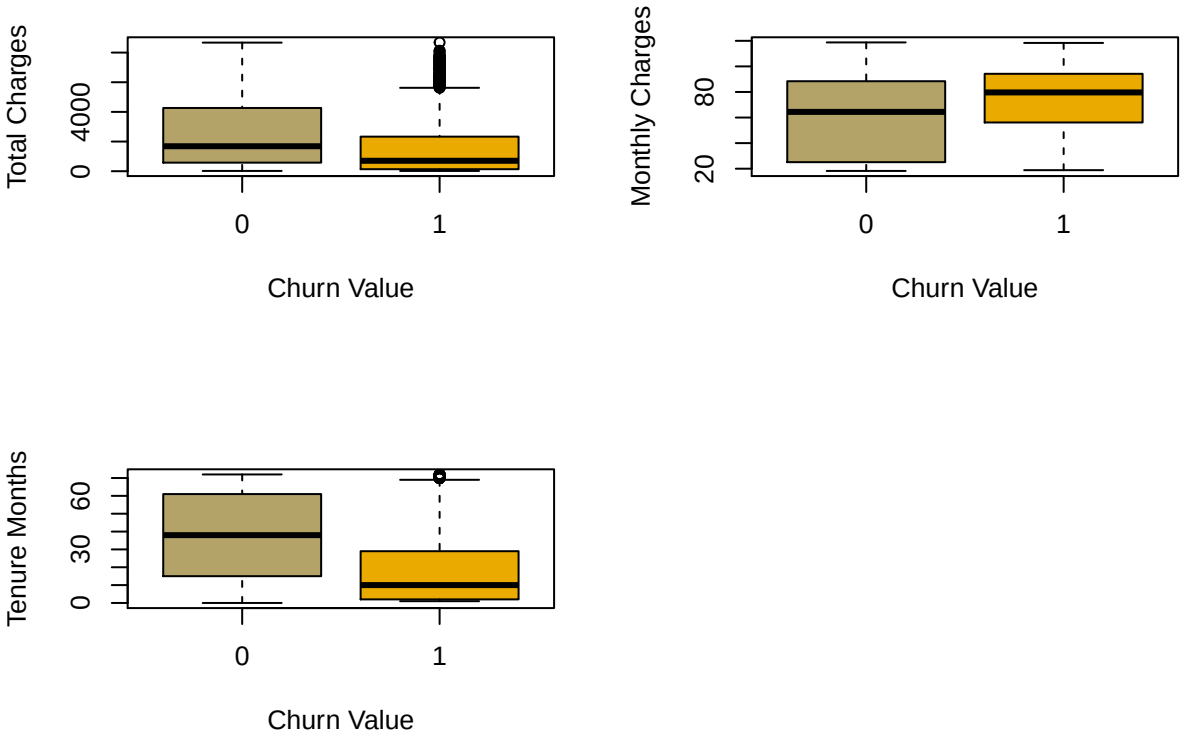
### General Data Prep

```
# Convert categorical variables to factors
dat <- dat %>% mutate_at(vars(-c(`Churn.Reason`,`Churn.Score`,`Churn.Value`,`Total.Charges`,`Monthly.Cha
```

## Exploratory Data Analysis (EDA)

### Relationship between binary response variable and numerical Variables

```
par(mfrow=c(2,2))
# Plot Churn Value against Total Charges
boxplot(Total.Charges ~ Churn.Value, main="", xlab="Churn Value",
        ylab="Total Charges", col=c(techgold,buzzgold), data=dat)

# Plot Churn Value against Monthly Charges
boxplot(Monthly.Charges ~ Churn.Value, main="", xlab="Churn Value",
        ylab="Monthly Charges", col=c(techgold,buzzgold), data=dat)

# Plot Churn Value against Tenure Months
boxplot(Tenure.Months ~ Churn.Value, main="", xlab="Churn Value",
        ylab="Tenure Months", col=c(techgold,buzzgold), data=dat)
```

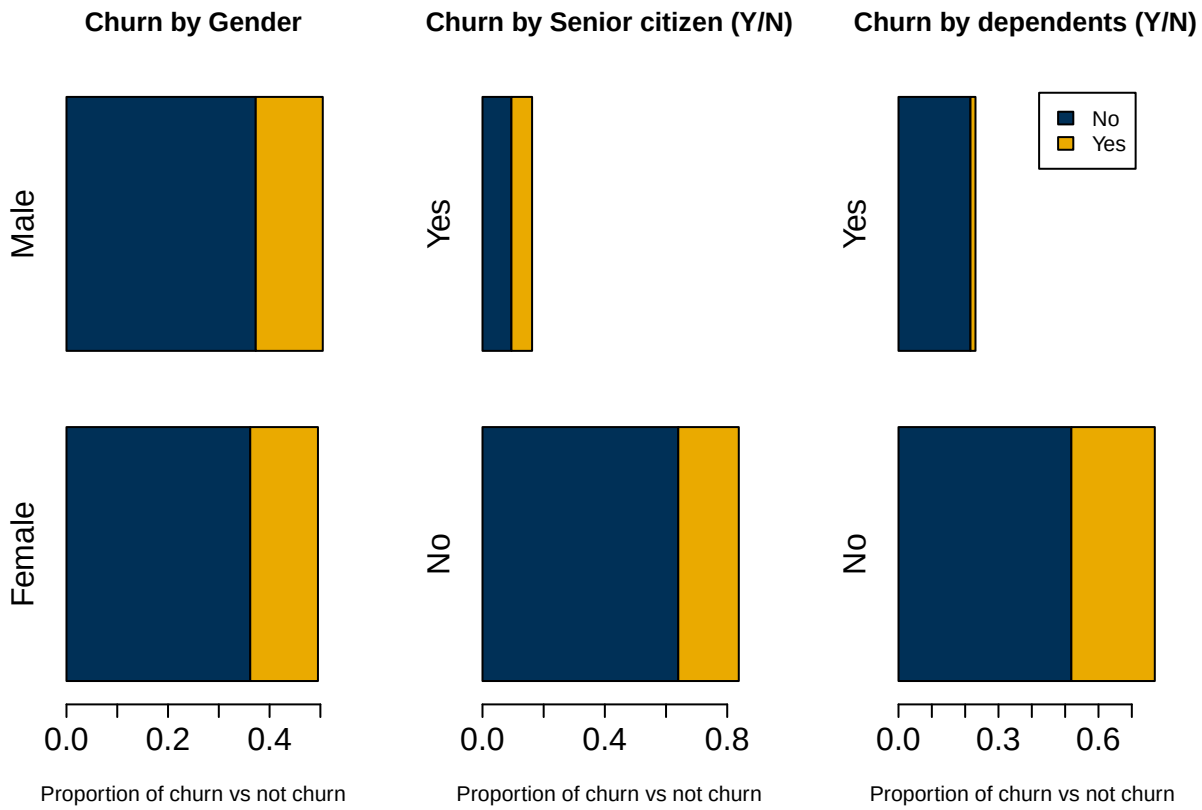



- There seems to exist differences in the total charges, monthly charges and tenure months between customers that have remained in the company versus customers that have left the company.
- Customers that have not churned appear to have higher total charges and tenure months but lower monthly charges than customers that have churned.

### Relationship between binary response variable and categorical variables

```
# Barplot Churn Value against Gender
par(mfrow=c(1,3))
tb_obgender = xtabs(~dat$Churn.Value+ dat$Gender)
barplot(prop.table(tb_obgender),axes=T,space=0.3,  cex.axis=1.5, cex.names=1.5,
        xlab="Proportion of churn vs not churn",
        horiz=T, col=c(gtblue,buzzgold),main="Churn by Gender")
```

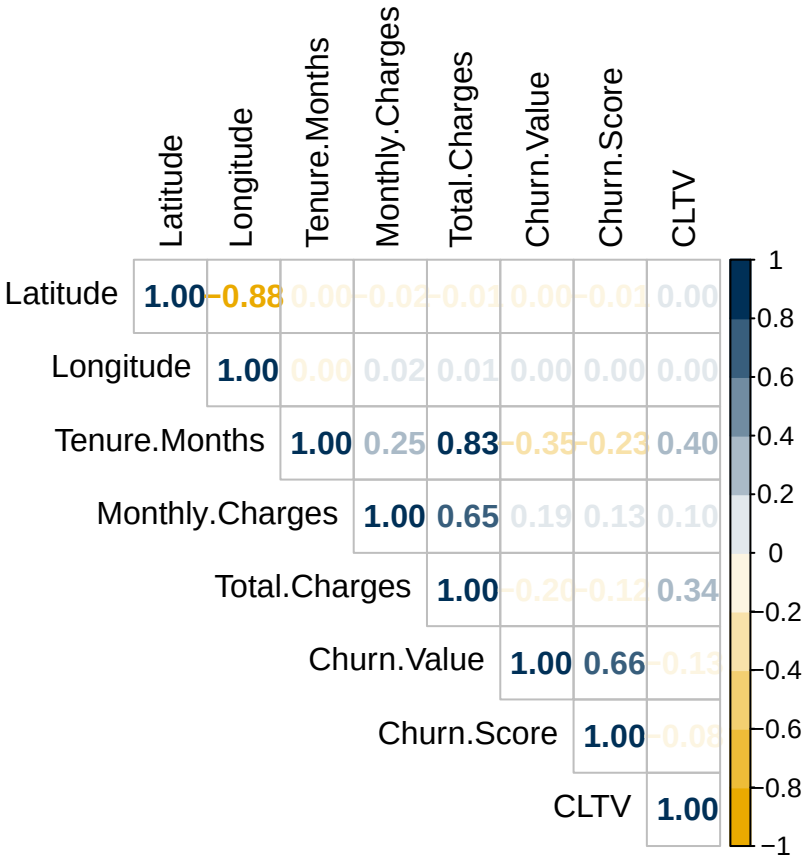
```r
# Barplot Churn Value against Senior Citizen
tb_citizen = xtabs(~dat$Churn.Value+ dat$Senior.Citizen)
barplot(prop.table(tb_citizen),axes=T,space=0.3, cex.axis=1.5, cex.names=1.5,
        xlab="Proportion of churn vs not churn",
        horiz=T, col=c(gtblue,buzzgold),main="Churn by Senior citizen (Y/N)")

# Barplot Churn Value against Dependents
tb_Dependents = xtabs(~dat$Churn.Value+ dat$Dependents)
barplot(prop.table(tb_Dependents),axes=T,space=0.3,cex.axis=1.5, cex.names=1.5,
        xlab="Proportion of churn vs not churn ",
        horiz=T, col=c(gtblue,buzzgold),main="Churn by dependents (Y/N)",
        legend.text = c("No", "Yes"))
```




There seems to exist significant differences in the proportions for each group in the predicting variables *Senior Citizen* and *Dependents*.

### Correlation among the numeric variables

```r
# Select numerical variables
dat.num <- na.omit(dat[ , which(sapply(dat, is.numeric))])
# Create correlation matrix
corr <- cor(dat.num)
# Create correlation plot
col <- colorRampPalette(c(buzzgold,"white", gtblue))(10)
par(mfrow=c(1,1))
corrplot(corr, method = "number", type = "upper", tl.col="black", col = col)
```




- There appears to be strong correlation among some of the predicting variables, particularly between the following pairs: *Total Charges-Tenure Months*, *Total Charges-Monthly Cahrges*, *Latitude-Longitude* and *Churn Value-Churn Score*. This suggests that multicollinearity could be a problem, and we should not include all the predictors in the logistic regression model.

## Logistic Regression and Model Selection

Now, we will build a logistic regression model for predicting which customers are likely to churn. We will explore Forward-Backward Stepwise Regression, lasso regression and elastic net regression for model selection.

### Data Preparation

```
# 1. Dropping columns not considered further in this analysis
drops <- c("Country", "State", "Churn.Label","CLTV", "Churn.Reason", "City", "Zip.Code", "Lat.Long","La
dat.class <- dat[ , !(names(dat) %in% drops)]

# 2. Dropping observations with missing values NA's
nas <- dat.class[rowSums(is.na(dat.class)) > 0,]
cat("There are", dim(nas)[1], "NA's in the data \n")
```

```
## There are 11 NA's in the data
```

```
dat.class  <- na.omit(dat.class)
cat("There are", dim(dat.class)[1], "observations in the reduced data")
```

```
## There are 7032 observations in the reduced data
```

```r
# 3. Removing redundant levels to avoid perfect collinearity between dummy variables
dat.class[dat.class == "No internet service" | dat.class == "No phone service" ] <- "No"
var_list <- c("Multiple.Lines","Online.Security", "Online.Backup",
              "Device.Protection", "Tech.Support", "Streaming.TV", "Streaming.Movies")
for(i in var_list){
  dat.class[[i]]<- droplevels(dat.class[[i]])
}

# 4. Data Split- 75% Train 25% Test
set.seed(1)
split = sample.split(dat.class$Churn.Value, SplitRatio = 0.75)
train = subset(dat.class, split == TRUE)
test = subset(dat.class, split == FALSE)
```

### Creating the Full Model

```r
# Building the model
full.model <- glm(Churn.Value ~ ., family = "binomial", data = train)
summary(full.model)
```

```
##
## Call:
## glm(formula = Churn.Value ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9009  -0.6593  -0.2566   0.6855   3.3948
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           7.068e-01  9.654e-01   0.732 0.464082
## GenderMale                           -2.476e-02  7.627e-02  -0.325 0.745484
## Senior.CitizenYes                     1.283e-01  9.667e-02   1.328 0.184296
## PartnerYes                            3.563e-01  8.762e-02   4.067 4.77e-05
## DependentsYes                        -1.517e+00  1.368e-01 -11.092  < 2e-16
## Tenure.Months                        -5.846e-02  7.272e-03  -8.039 9.05e-16
## Phone.ServiceYes                     -1.203e-01  7.712e-01  -0.156 0.876016
## Multiple.LinesYes                     3.281e-01  2.095e-01   1.567 0.117196
## Internet.ServiceFiber optic           1.220e+00  9.463e-01   1.289 0.197348
## Internet.ServiceNo                   -1.300e+00  9.584e-01  -1.357 0.174913
## Online.SecurityYes                   -1.893e-01  2.102e-01  -0.901 0.367816
## Online.BackupYes                      4.193e-02  2.075e-01   0.202 0.839814
## Device.ProtectionYes                  2.577e-02  2.092e-01   0.123 0.901969
## Tech.SupportYes                      -2.576e-01  2.134e-01  -1.207 0.227541
## Streaming.TVYes                       4.968e-01  3.871e-01   1.283 0.199383
## Streaming.MoviesYes                   4.681e-01  3.884e-01   1.205 0.228187
## ContractOne year                     -7.340e-01  1.260e-01  -5.825 5.70e-09
## ContractTwo year                     -1.452e+00  2.114e-01  -6.868 6.52e-12
## Paperless.BillingYes                  3.895e-01  8.785e-02   4.433 9.28e-06
## Payment.MethodCredit card (automatic) -1.244e-01  1.343e-01  -0.926 0.354418
## Payment.MethodElectronic check        2.829e-01  1.107e-01   2.556 0.010591
## Payment.MethodMailed check            6.203e-03  1.347e-01   0.046 0.963263
## Monthly.Charges                      -2.287e-02  3.768e-02  -0.607 0.543852
```

```
## Total.Charges                           2.875e-04  8.288e-05   3.469 0.000522
##
## (Intercept)
## GenderMale
## Senior.CitizenYes
## PartnerYes                            ***
## DependentsYes                         ***
## Tenure.Months                         ***
## Phone.ServiceYes
## Multiple.LinesYes
## Internet.ServiceFiber optic
## Internet.ServiceNo
## Online.SecurityYes
## Online.BackupYes
## Device.ProtectionYes
## Tech.SupportYes
## Streaming.TVYes
## Streaming.MoviesYes
## ContractOne year                      ***
## ContractTwo year                      ***
## Paperless.BillingYes                  ***
## Payment.MethodCredit card (automatic)
## Payment.MethodElectronic check        *
## Payment.MethodMailed check
## Monthly.Charges
## Total.Charges                         ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6108.0  on 5273  degrees of freedom
## Residual deviance: 4223.2  on 5250  degrees of freedom
## AIC: 4271.2
##
## Number of Fisher Scoring iterations: 6
```

### Finding insignificant Variables

```
# Coefficients
which(summary(full.model)$coeff[,4]>0.05)
```

```
##                           (Intercept)                            GenderMale
##                                     1                                     2
##                      Senior.CitizenYes                     Phone.ServiceYes
##                                     3                                     7
##                     Multiple.LinesYes           Internet.ServiceFiber optic
##                                     8                                     9
##                    Internet.ServiceNo                   Online.SecurityYes
##                                    10                                    11
##                      Online.BackupYes                 Device.ProtectionYes
##                                    12                                    13
##                     Tech.SupportYes                     Streaming.TVYes
##                                    14                                    15
```

```
##                   Streaming.MoviesYes Payment.MethodCredit card (automatic)
##                                    16                                    20
##            Payment.MethodMailed check                         Monthly.Charges
##                                    22                                    23
```

**Test for overall regression**

```
# Overall Significance
gstat = full.model$null.deviance - deviance(full.model)
pvalue = 1-pchisq(gstat,length(coef(full.model))-1)
cbind(gstat, pvalue)
```

```
##        gstat pvalue
## [1,] 1884.79      0
```

The p-value is very close to zero, so we can reject $H_0 : \beta_1 = ... = \beta_p = 0$. The overall regression appears to have explanatory power.

**Model Assessment**

1. Multicollinearity

```
# Calculating GVIF
vifs <- vif(full.model)
vifs
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## Gender                 1.003414  1        1.001705
## Senior.Citizen         1.112401  1        1.054704
## Partner                1.248636  1        1.117424
## Dependents             1.098666  1        1.048173
## Tenure.Months         15.612548  1        3.951272
## Phone.Service         35.526189  1        5.960385
## Multiple.Lines         7.434935  1        2.726708
## Internet.Service     382.924211  2        4.423624
## Online.Security        5.158636  1        2.271263
## Online.Backup          6.520493  1        2.553526
## Device.Protection      6.611606  1        2.571304
## Tech.Support           5.409603  1        2.325855
## Streaming.TV          25.075402  1        5.007534
## Streaming.Movies      25.317771  1        5.031677
## Contract               1.625406  2        1.129121
## Paperless.Billing      1.128532  1        1.062324
## Payment.Method         1.413278  3        1.059346
## Monthly.Charges      694.903171  1       26.361016
## Total.Charges         20.166529  1        4.490716
```

As GVIFs of some variables are high, it indicates that there is a problem of multicollinearity in the model.

**Variable Selection**

1. Forward-Backward Stepwise Regression

```
# Create minimum model including an intercept
min.model <-  glm(Churn.Value~ 1, family = "binomial", data = train)

# Perform stepwise regression
```

```
step.model <- step(min.model, scope = list(lower = min.model, upper = full.model),
                   direction = "both", trace = FALSE)
summary(step.model)
```

Call: glm(formula = Churn.Value ~ Contract + Internet.Service + Tenure.Months + Dependents + Streaming.TV + Paperless.Billing + Payment.Method + Partner + Tech.Support + Streaming.Movies + Online.Security + Total.Charges + Phone.Service + Multiple.Lines, family = "binomial", data = train)

Deviance Residuals: Min 1Q Median 3Q Max
-1.8534 -0.6570 -0.2563 0.6886 3.3899

Coefficients: Estimate Std. Error z value Pr(>|z|) (Intercept) 9.350e-02 1.937e-01 0.483 0.629294 ContractOne year -7.538e-01 1.254e-01 -6.012 1.83e-09 ContractTwo year -1.479e+00 2.106e-01 -7.022 2.19e-12 Internet.ServiceFiber optic 6.731e-01 1.157e-01 5.820 5.88e-09 Internet.ServiceNo -7.022e-01 1.617e-01 -4.344 1.40e-05 Tenure.Months -5.721e-02 7.208e-03 -7.937 2.07e-15 DependentsYes -1.540e+00 1.356e-01 -11.352 < 2e-16 Streaming.TVYes 2.631e-01 9.527e-02 2.761 0.005758 Paperless.BillingYes 3.981e-01 8.767e-02 4.541 5.60e-06 Payment.MethodCredit card (automatic) -1.277e-01 1.342e-01 -0.951 0.341458 Payment.MethodElectronic check 2.892e-01 1.106e-01 2.616 0.008905 Payment.MethodMailed check 5.494e-03 1.344e-01 0.041 0.967386 PartnerYes 3.631e-01 8.722e-02 4.163 3.14e-05 Tech.SupportYes -3.852e-01 1.006e-01 -3.830 0.000128 Streaming.MoviesYes 2.385e-01 9.548e-02 2.498 0.012492 Online.SecurityYes -3.028e-01 9.910e-02 -3.055 0.002248 Total.Charges 2.640e-04 8.059e-05 3.276 0.001053 Phone.ServiceYes -5.755e-01 1.557e-01 -3.697 0.000218 Multiple.LinesYes 2.227e-01 9.389e-02 2.373 0.017668

(Intercept)
ContractOne year ***ContractTwo year*** Internet.ServiceFiber optic ***Internet.ServiceNo*** Tenure.Months ***DependentsYes*** Streaming.TVYes ** Paperless.BillingYes ***Payment.MethodCredit card (automatic)*** ***Payment.MethodElectronic check*** *Payment.MethodMailed check*
*PartnerYes* **Tech.SupportYes** *Streaming.MoviesYes*
Online.SecurityYes ** Total.Charges ** Phone.ServiceYes ** *Multiple.LinesYes*
— Signif. codes: 0 '' ***0.001*** '' *0.01* '' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 6108  on 5273  degrees of freedom
```

Residual deviance: 4227 on 5255 degrees of freedom AIC: 4265

Number of Fisher Scoring iterations: 6

```
which(summary(step.model)$coeff[,4]>0.05)
```

```
                    (Intercept) Payment.MethodCredit card (automatic)
                              1                                     10
      Payment.MethodMailed check
                             12
```

2. Lasso Regression

```
# Set a seed for reproducibility
set.seed(1)

# Set predictors and response to correct format
x.train <- model.matrix(Churn.Value ~ ., train)[,-1]
y.train <- train$Churn.Value

# Use cross validation to find optimal lambda
cv.lasso <- cv.glmnet(x.train, y.train, alpha = 1, family = "binomial")

# Train Lasso and display coefficients with optimal lambda
```

```r
lasso.model <- glmnet(x.train, y.train, alpha = 1, family = "binomial")
coef(lasso.model, cv.lasso$lambda.min)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                                                s1
## (Intercept)                          0.0616908714
## GenderMale                          -0.0189827491
## Senior.CitizenYes                    0.1267094010
## PartnerYes                           0.3433505269
## DependentsYes                       -1.5015533718
## Tenure.Months                       -0.0533519875
## Phone.ServiceYes                    -0.5387783363
## Multiple.LinesYes                    0.2155879341
## Internet.ServiceFiber optic          0.6774842373
## Internet.ServiceNo                  -0.7153865509
## Online.SecurityYes                  -0.2901886445
## Online.BackupYes                    -0.0551844611
## Device.ProtectionYes                -0.0713237269
## Tech.SupportYes                     -0.3586308465
## Streaming.TVYes                      0.2764996042
## Streaming.MoviesYes                  0.2469947909
## ContractOne year                    -0.7264637316
## ContractTwo year                    -1.4195800593
## Paperless.BillingYes                 0.3846621649
## Payment.MethodCredit card (automatic) -0.1250209754
## Payment.MethodElectronic check       0.2799501370
## Payment.MethodMailed check           .
## Monthly.Charges                      .
## Total.Charges                        0.0002267873
```

3. Elastic Net Regression

```r
# Set a seed for reproducibility
set.seed(1)

# Use cross validation to find optimal lambda
cv.elnet <- cv.glmnet(x.train, y.train, alpha = 0.5, family = "binomial")

# Train Elastic Net and display coefficients with optimal lambda
elnet.model <- glmnet(x.train, y.train, alpha = 0.5, family = "binomial")
coef(elnet.model, cv.elnet$lambda.min)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                                                s1
## (Intercept)                          0.0632467208
## GenderMale                          -0.0218748195
## Senior.CitizenYes                    0.1275969043
## PartnerYes                           0.3465187468
## DependentsYes                       -1.5044928078
## Tenure.Months                       -0.0532672883
## Phone.ServiceYes                    -0.5440255046
## Multiple.LinesYes                    0.2175833824
## Internet.ServiceFiber optic          0.6773960404
## Internet.ServiceNo                  -0.7200792206
## Online.SecurityYes                  -0.2932546962
```

```
## Online.BackupYes                       -0.0594417946
## Device.ProtectionYes                   -0.0758406747
## Tech.SupportYes                        -0.3612297458
## Streaming.TVYes                         0.2783509745
## Streaming.MoviesYes                     0.2489511586
## ContractOne year                       -0.7304546600
## ContractTwo year                       -1.4248500893
## Paperless.BillingYes                    0.3858558449
## Payment.MethodCredit card (automatic) -0.1250267922
## Payment.MethodElectronic check          0.2829523057
## Payment.MethodMailed check              0.0075598117
## Monthly.Charges                         .
## Total.Charges                           0.0002271914
```

4. Variable Selection Comparison

```
# Identify variables not selected by F-B Stepwise regression
index.step <- which(!(names(coef(full.model)) %in% names(coef(step.model))))
cat("\n Variables not selected by forward-backward stepwise:",
    names(coef(full.model)[index.step]))
```

```
##
##  Variables not selected by forward-backward stepwise: GenderMale Senior.CitizenYes Online.BackupYes
```

```
# Identify variables not selected by Lasso
index.lasso <- which(coef(lasso.model, cv.lasso$lambda.min) == 0)
cat("\n Variables not selected by lasso regression: ",
    names(coef(full.model)[index.lasso]))
```

```
##
##  Variables not selected by lasso regression:  Payment.MethodMailed check Monthly.Charges
```

```
# Identify variables not selected by Elastic Net
index.elnet <- which(coef(elnet.model, cv.elnet$lambda.min) == 0)
cat("\n Variables not selected by elastic net regression:",
    names(coef(full.model)[index.elnet]))
```

```
##
##  Variables not selected by elastic net regression: Monthly.Charges
```

### Prediction on Test Set

Now, we are on to do the predictions using the models we just created. A classification threshold of 0.5 is used. Note that this threshold could be tuned depending on the sensitivity/specificity tolerance. In this case, it becomes important to identify people that are likely to churn so that the corrective measures can be taken. This means lowering the threshold could be a good idea even if it results in more False Positive cases.

```
# 1. Prediction for the full model

# Obtain predicted probabilities for the test set
pred.full = predict(full.model, newdata = test, type = "response")
# Obtain classifications using a classification threshold of 0.5
predClass.full = ifelse(pred.full > 0.5, 1, 0)

# 2. Prediction for the stepwise regression

# Obtain predicted probabilities for the test set
pred.step = predict(step.model, newdata = test, type = "response")
```

```
# Obtain classifications using a classification threshold of 0.5
predClass.step = ifelse(pred.step > 0.5, 1, 0)

# 3. Prediction for the lasso regression

# Retrain OLS model using Lasso-selected predictors
lasso.predictors <- as.data.frame(x.train)[-(index.lasso-1)]
lasso.retrained <- glm(y.train ~ . , family = "binomial", data = lasso.predictors)
# Set test data to correct format
new_test <- model.matrix( ~ ., test)[,-1]
# Obtain predicted probabilities for the test set
pred.lasso = predict(lasso.retrained, newdata = as.data.frame(new_test),
                     type = "response")
# Obtain classifications using a classification threshold of 0.5
predClass.lasso = ifelse(pred.lasso > 0.5, 1, 0)

# 4. Prediction for elastic net regression

# Set predictors to correct format
x.test <- model.matrix(Churn.Value ~ ., test)[,-1]
# Obtain predicted probabilities for the test set
pred.elnet = as.vector(predict(elnet.model, newx = x.test,
                               type = "response", s = cv.elnet$lambda.min))
# Obtain classifications using a classification threshold of 0.5
predClass.elnet = ifelse(pred.elnet > 0.5, 1, 0)

# Create a data frame with the predictions
preds = data.frame(ChurnValue = test$Churn.Value, predClass.full,
                   predClass.step, predClass.lasso, predClass.elnet)
```

### Classification Evaluation Metrics

Now we could see how the models are doing when predicting on a new set of data. We will calculate the Accuracy, the Sensitivity (true positive rate) and the Specificity (false positive rate) metrics to evaluate these models at 0.5 threshold.

```
# Build a confusion table and calculate metrics
pred_metrics = function(modelName, actualClass, predClass) {
  cat(modelName, '\n')
  conmat <- confusionMatrix(table(actualClass, predClass))
  c(conmat$overall["Accuracy"], conmat$byClass["Sensitivity"],
    conmat$byClass["Specificity"])
}

pred_metrics("Full Model",test$Churn.Value, predClass.full)
```

```
## Full Model

##    Accuracy Sensitivity Specificity
##   0.8179750   0.8577745   0.6832918
```

```
pred_metrics("Stepwise Regression Model",test$Churn.Value, predClass.step)
```

```
## Stepwise Regression Model

##    Accuracy Sensitivity Specificity
```

```
##   0.8174061   0.8581979   0.6806931
```

```
pred_metrics("Lasso Regression Model",test$Churn.Value, predClass.lasso)
```

```
## Lasso Regression Model
```

```
##    Accuracy Sensitivity Specificity
##   0.8174061   0.8576696   0.6815920
```

```
pred_metrics("Elastic Regression Model",test$Churn.Value, predClass.elnet)
```

```
## Elastic Regression Model
```

```
##    Accuracy Sensitivity Specificity
##   0.8179750   0.8572480   0.6842105
```

All models have very similar prediction metrics. In this case, we are concerned about identifying the customers that are likely to churn. Since correctly identifying positives is important for us, then we should choose a model with higher Sensitivity. The model that includes the variables selected by Stepwise Regression has the highest sensitivity and is the model with the smallest amount of variables.

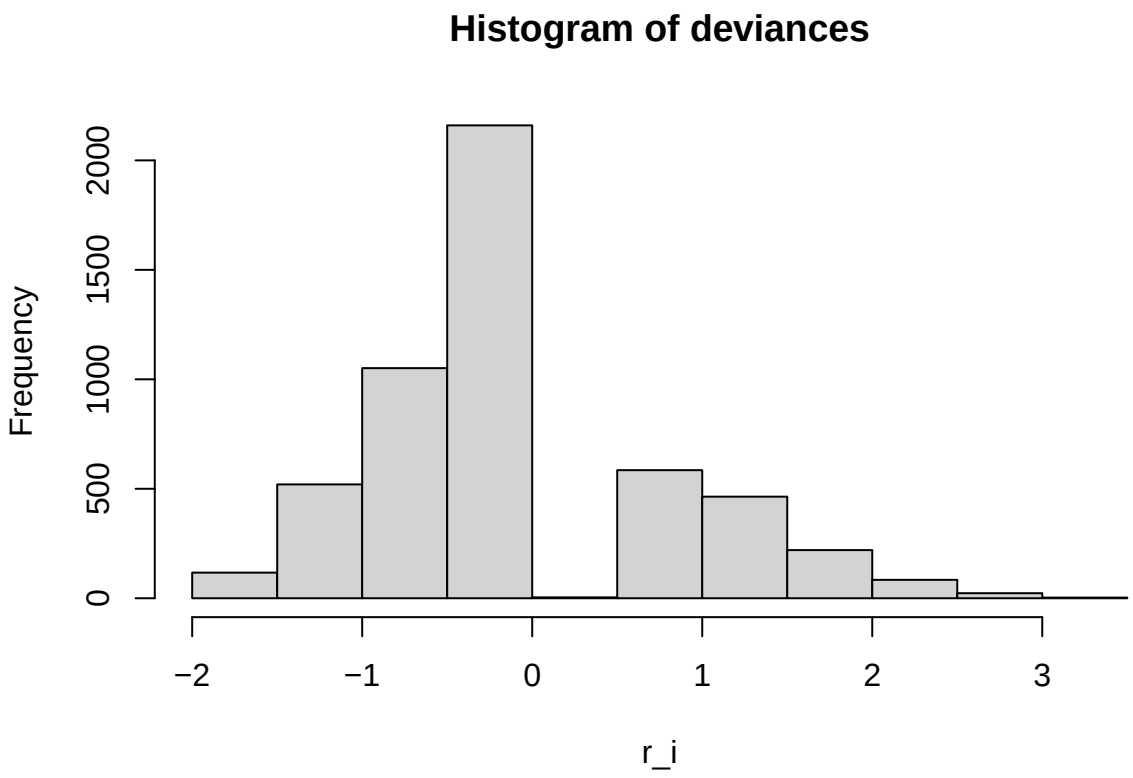
### Goodness of Fit

```
# Removing variables not selected by stepwise regression
step.predictors <-  names(coef(full.model)[index.step])
x.train <- as.data.frame(x.train)
train.final <- x.train[, - which(colnames(x.train) %in% step.predictors)]

# Aggregating the data
obdata.agg.n = aggregate(y.train ~ . , data = train.final, FUN = length)
obdata.agg.y = aggregate(y.train ~ . , data = train.final, FUN = sum)
dat.aggr <- cbind(obdata.agg.y, total = obdata.agg.n$y.train)

## Fitting the model
mod.aggr = glm(y.train / total ~ . , data = dat.aggr, weight = total, family = binomial)
summary(mod.aggr)
```

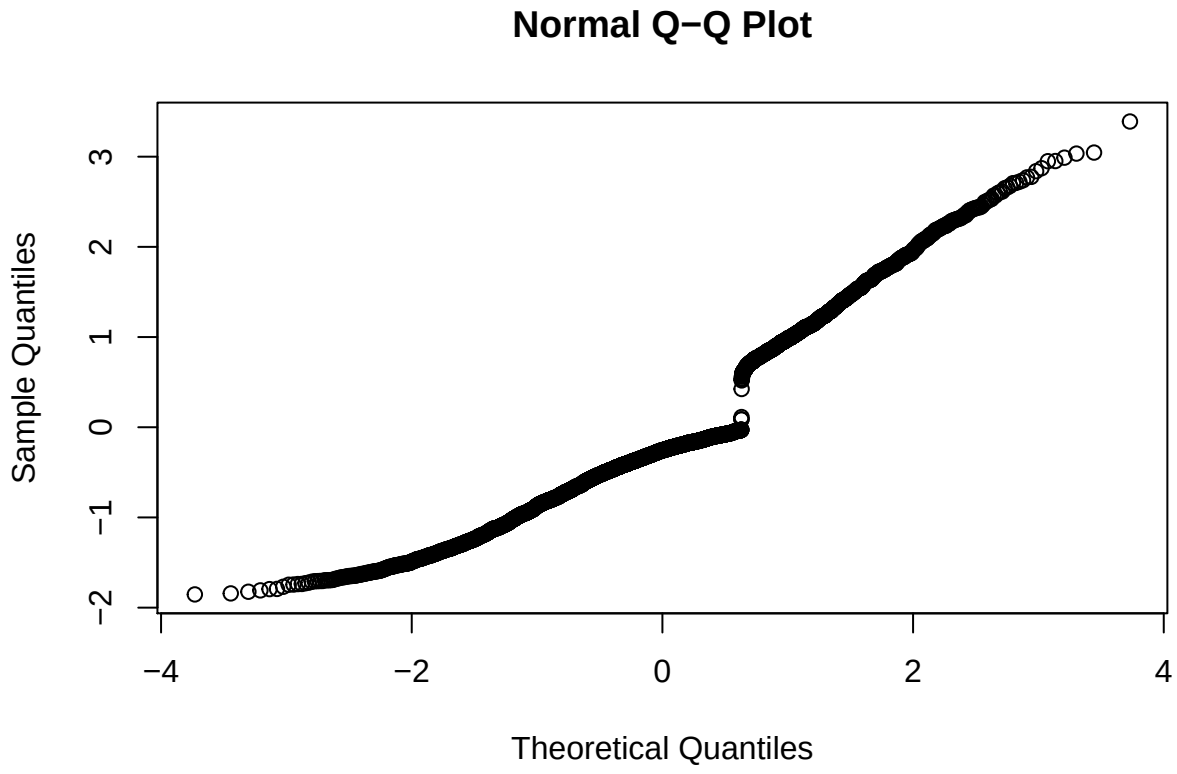
```
##
## Call:
## glm(formula = y.train/total ~ ., family = binomial, data = dat.aggr,
##     weights = total)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8534  -0.6523  -0.2550   0.6785   3.3899
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          9.350e-02  1.937e-01   0.483 0.629294
## PartnerYes                           3.631e-01  8.722e-02   4.163 3.14e-05
## DependentsYes                       -1.540e+00  1.356e-01 -11.352  < 2e-16
## Tenure.Months                       -5.721e-02  7.208e-03  -7.937 2.07e-15
## Phone.ServiceYes                    -5.755e-01  1.557e-01  -3.697 0.000218
## Multiple.LinesYes                    2.227e-01  9.389e-02   2.373 0.017668
## `Internet.ServiceFiber optic`        6.731e-01  1.157e-01   5.820 5.88e-09
## Internet.ServiceNo                  -7.022e-01  1.617e-01  -4.344 1.40e-05
## Online.SecurityYes                  -3.028e-01  9.910e-02  -3.055 0.002248
```

```
## Tech.SupportYes                          -3.852e-01  1.006e-01  -3.830 0.000128
## Streaming.TVYes                           2.631e-01  9.527e-02   2.761 0.005758
## Streaming.MoviesYes                       2.385e-01  9.548e-02   2.498 0.012492
## `ContractOne year`                       -7.538e-01  1.254e-01  -6.012 1.83e-09
## `ContractTwo year`                       -1.479e+00  2.106e-01  -7.022 2.19e-12
## Paperless.BillingYes                      3.981e-01  8.767e-02   4.541 5.60e-06
## `Payment.MethodCredit card (automatic)` -1.277e-01  1.342e-01  -0.951 0.341458
## `Payment.MethodElectronic check`          2.892e-01  1.106e-01   2.616 0.008905
## `Payment.MethodMailed check`              5.494e-03  1.344e-01   0.041 0.967386
## Total.Charges                             2.640e-04  8.059e-05   3.276 0.001053
##
## (Intercept)
## PartnerYes                               ***
## DependentsYes                            ***
## Tenure.Months                            ***
## Phone.ServiceYes                         ***
## Multiple.LinesYes                        *
## `Internet.ServiceFiber optic`            ***
## Internet.ServiceNo                       ***
## Online.SecurityYes                       **
## Tech.SupportYes                          ***
## Streaming.TVYes                          **
## Streaming.MoviesYes                      *
## `ContractOne year`                       ***
## `ContractTwo year`                       ***
## Paperless.BillingYes                     ***
## `Payment.MethodCredit card (automatic)`
## `Payment.MethodElectronic check`         **
## `Payment.MethodMailed check`
## Total.Charges                            **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6061.6  on 5230  degrees of freedom
## Residual deviance: 4180.5  on 5212  degrees of freedom
## AIC: 4239.5
##
## Number of Fisher Scoring iterations: 6
```

```r
# Test for GOF : Using deviance residuals and check normality of deviance residuals
res = resid(mod.aggr, type="deviance")
hist(res, main="Histogram of deviances",breaks = 8,xlab = "r_i")
```




```
qqnorm(res)
```




```
cbind(statistic = sum(res^2), pvalue = 1-pchisq(sum(res^2), mod.aggr$df.resid))
```

```
##      statistic pvalue
```

```
## [1,]  4180.503      1
```

- P-value is high, thus our model reasonably fits the data according to this test.

## K-Nearest Neighbor (KNN) Classification

Next, we will use K-Nearest Neighbor (KNN) for predicting which customers are likely to churn. We will look at different kernels to measure the distance between neighbors: rectangular (which is standard unweighted knn), triangular, epanechnikov, and optimal.

### Data Prep

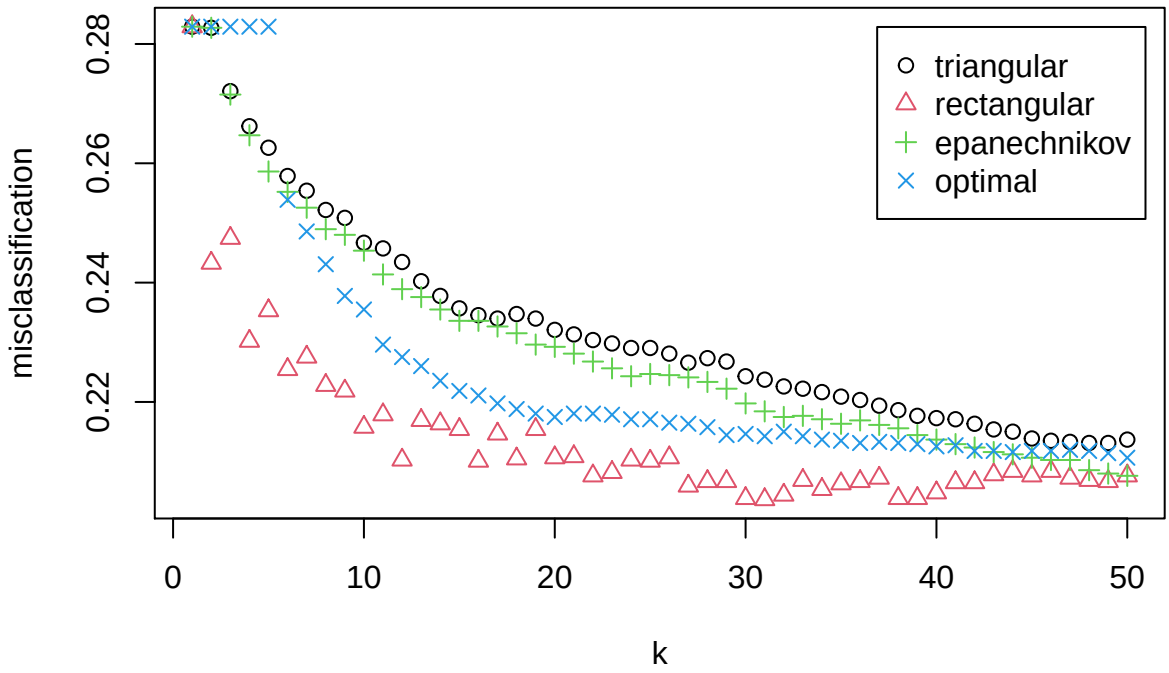
```
# Convert response to factor
y.train <- as.factor(train$Churn.Value)
y.test <- as.factor(test$Churn.Value)

# Dummify categorical features
dummies.train <- dummyVars(Churn.Value ~ ., data = train)
dummies.test <- dummyVars(Churn.Value ~ ., data = test)

# Create data frames containing the predictors
x.train.knn <- data.frame(predict(dummies.train, newdata = train))
x.test.knn <- data.frame(predict(dummies.test, newdata = test))
```

### Train the KNN model

```
# Set a seed for reproducibility
set.seed (1)

# Use leave-one-out cross-validation to find the optimal value of "k"
(kknn.train <- train.kknn(y.train ~ ., x.train.knn, kmax = 50,
                          kernel = c("triangular", "rectangular",
                                     "epanechnikov", "optimal"),
                          scale = TRUE))
```

```
##
## Call:
## train.kknn(formula = y.train ~ ., data = x.train.knn, kmax = 50,     kernel = c("triangular", "recta
##
## Type of response variable: nominal
## Minimal misclassification: 0.2036405
## Best kernel: rectangular
## Best k: 31
```

```
# Plot of missclasfication errors vs. k for different kernels
plot(kknn.train)
```




```
# Extracting optimal kernel and k
cat("\n The lowest missclassification error is achieved with a",
    kknn.train$best.parameters[[1]],
    "kernel and a number of nearest neighbors (k) of",
    kknn.train$best.parameters[[2]])
```

```
##
##  The lowest missclassification error is achieved with a rectangular kernel and a number of nearest n
```

### Prediction and Model Evaluation on Test Set

```
# Predict the labels on the test set
pred.knn <- predict(kknn.train, x.test.knn)

#  Calculate classification Evaluation Metrics
pred_metrics("KNN", y.test, pred.knn)
```

```
## KNN
```

```
##    Accuracy Sensitivity Specificity
##   0.7957907   0.8606811   0.6158798
```

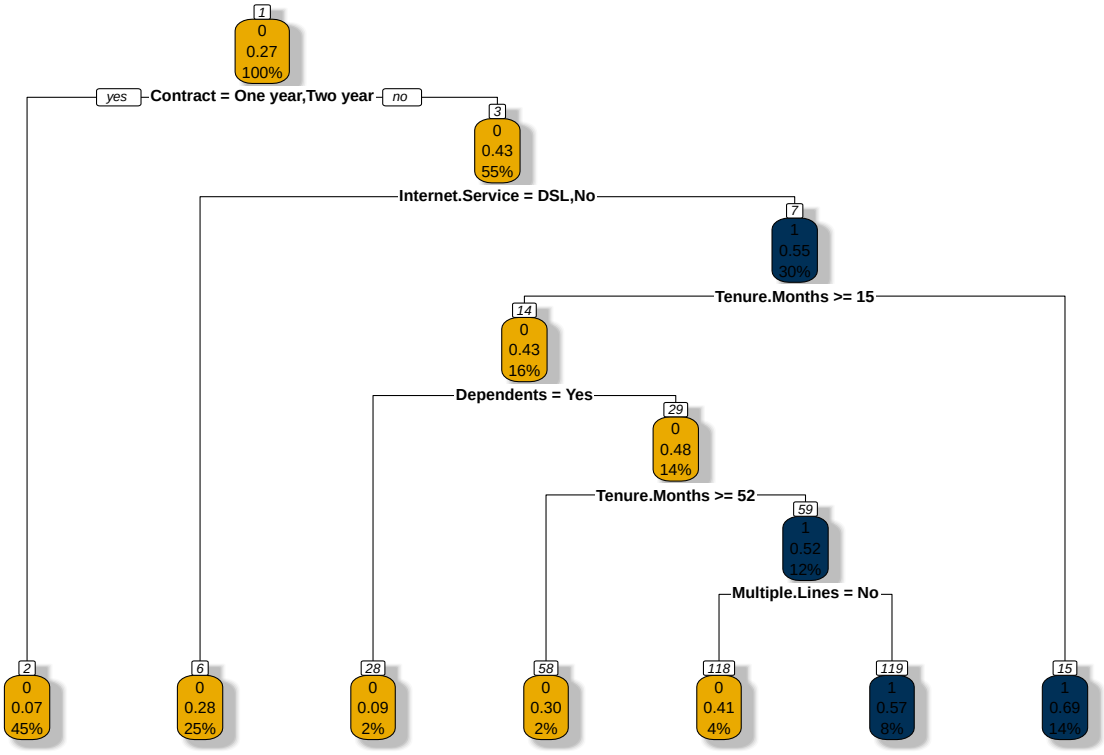
## Decision Tree and Random Forest

In this section, we will use tree-based models such as decision tree and random forest for predicting which customers are likely to churn. In order to compare the predictions from both models, we will be using measures such as accuracy, sensitivity, and specificity.

### Decision Tree

Initially, we will remove the unnecessary spaces from the variable names, build our decision tree model, and plot it, to take a look at the tree structure and its splits.

```
#1. Removing the spaces from variable names
names(train)<-str_replace_all(names(train), c(" " = "." ))
names(test)<-str_replace_all(names(test), c(" " = "." ))


#2. Building model
set.seed(300)
decision_tree <- rpart(Churn.Value~., data = train, method = "class")

#3. Plotting model
rpart.plot(decision_tree, box.palette = c(buzzgold, gtblue), shadow.col = "gray", nn=TRUE)
```




In order to understand how the different probabilities of churn change for different customer and product features, we can analyze the decision tree model built above. The results are presented in the tree plot. The first number in the node corresponds to the classification of the node (0 if not churn and 1 if churn). The second number in the node corresponds to the predicted probability of churn. The third value in the node measures the total % of customers that are included in that node.

The most important insights from this plot are:

- The most important variable in determining churn rate is duration of contract. If the contract is 1 or 2 years the probability they will not churn is 93
- If the customer has a month-to-month contract, has fiber optic, is in default for more than 15 months and has dependents, the probability it will churn is only 9
- The higher churn occurs for month-to-month contracts, fiber optic, tenure higher than 15 months but lower than 52 months, no depedents and multiple lines. In that case, churn rate is 57
- Overall, the probabilities of churn are high for month-to-month contracts. The company can create incentives for customers to subscribe to longer contracts.

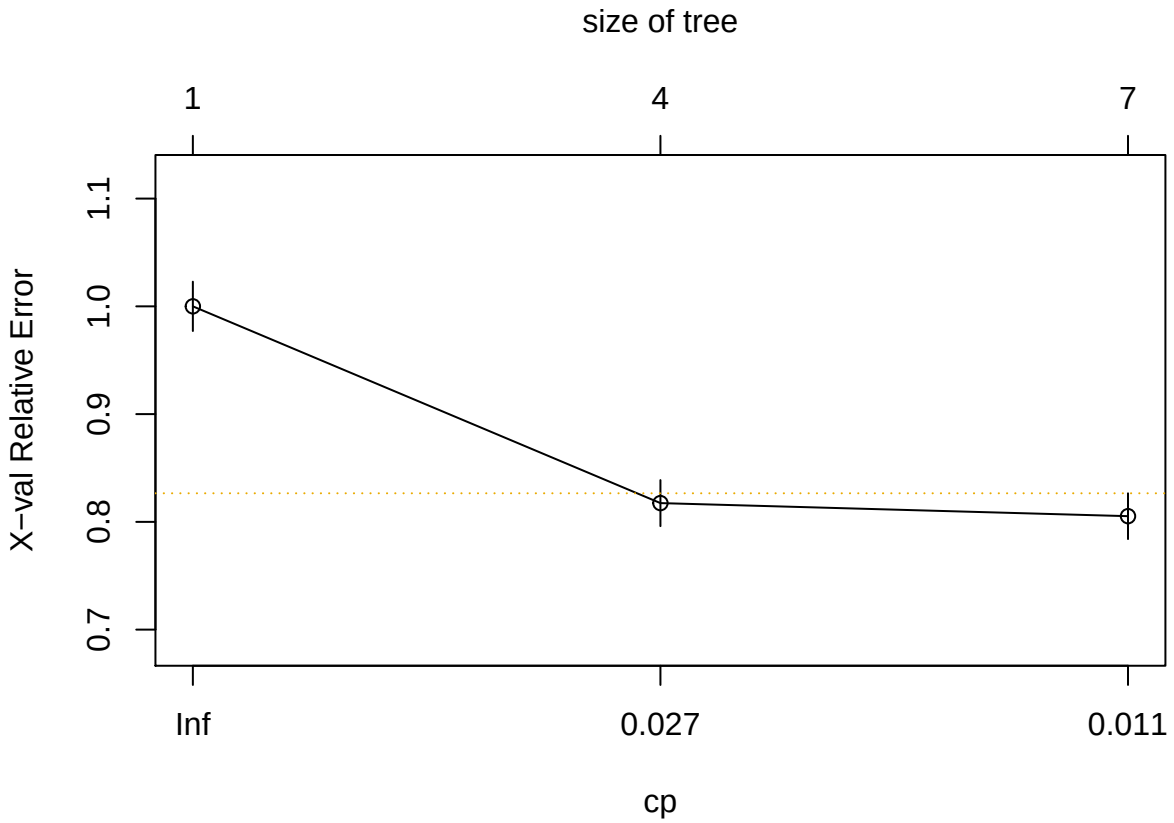
Now, since we have already fitted and plotted the model, we will use cross-validation using plotcp (function

of the rpart package) the complexity parameter table for the decision tree fit. This function will give us a visual representation of the cross-validation results in the decision tree object we just created.

```
#4. Visualize cross-validation in table format
printcp(decision_tree)
```

```
##
## Classification tree:
## rpart(formula = Churn.Value ~ ., data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] Contract         Dependents       Internet.Service Multiple.Lines
## [5] Tenure.Months
##
## Root node error: 1402/5274 = 0.26583
##
## n= 5274
##
##         CP nsplit rel error  xerror     xstd
## 1 0.056705      0   1.00000 1.00000 0.022884
## 2 0.013077      3   0.80029 0.81740 0.021362
## 3 0.010000      6   0.76106 0.80528 0.021247
```

```
#5. Plot the complexity parameter table
plotcp(decision_tree,minline = TRUE, lty = 3,col = buzzgold)
```




The table and plot contain the mean and standard deviation of the errors in the cross-validated prediction against each of the geometric means, which are displayed above. A good choice of cp for pruning is often the leftmost value for which the mean lies below the horizontal line. The main goal here is to control the size of the decision tree and to select the optimal tree size. If the cost of adding another variable to the decision tree

from the current node is above the value of cp, then tree building does not continue. We could also say that tree construction does not continue unless it would decrease the overall lack of fit by a factor of cp. As we noticed above, our tree had 7 variables and that is the optimal size, confirmed by the cp plot.

Since our tree is already in the optimal size, there is no need to prune it. However, in case we need to prune it, rpart package also contains a function where you reference the tree model and the optimal cp number, which in this case is 0.027 as you can see in the code below:

```
#6. Prune the tree model
pruned_tree <- prune(decision_tree,0.027)

#7. Predict Churn Score
predicted_churn_score <- predict(pruned_tree,test,type="class")

#8. Create Confusion Matrix
confusionMatrix(data = as.factor(predicted_churn_score), reference = as.factor(test$Churn.Value), positi
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1220  284
##          1   71  183
##
##                Accuracy : 0.7981
##                  95% CI : (0.7785, 0.8166)
##     No Information Rate : 0.7344
##     P-Value [Acc > NIR] : 2.967e-10
##
##                   Kappa : 0.3943
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3919
##             Specificity : 0.9450
##          Pos Pred Value : 0.7205
##          Neg Pred Value : 0.8112
##              Prevalence : 0.2656
##          Detection Rate : 0.1041
##    Detection Prevalence : 0.1445
##       Balanced Accuracy : 0.6684
##
##        'Positive' Class : 1
##
```

Above we have used our test data to make predictions using our decision tree model. We then classified the prediction results into 0 or 1 with type = "class", and finally created a confusion matrix to analyze the results and accuracy of our predictions. As we can see above, we have true/false positives/negatives, and on the bottom, accuracy, sensitivity, and specificity are also shown.

### More Information

Keep in mind that decision tree models can be optimiized in several different ways. Rpart is only one algorithm to be used in this scenario, there are many more such as tree, C5.0, and many others. For more information on those, look ar R documentation. Also it is important to know that there are many ways to optimize decision tree models. One way is using the train() function to select your desired algorithm, and automatically tune parameters. Other common ways are using bagged trees or boosting method, with those,

it is possible to automatically and manually tune parameters for better model performance based on your dataset. Please take a look on documentation or books such as "Machine Learning with R" by Brett Lantz.

### Random Forest

Now we will explore how to build a random forest model. While a decision tree is built on an entire dataset, using all the features/variables of interest, a random forest model selects observations/rows and specific features/variables to build multiple decision trees from, and then averages the results. This way, random forests are a stronger modeling technique and more robust than a single decision tree. Since they aggregate many decision trees, they limit overfitting, as well as error due to bias and therefore can produce better results.

```
set.seed(300)

# 1. Build Random Forest Model
rf <- randomForest(factor(Churn.Value)~.,data = train)

# 2. Predict using Random Forest Model
pred_test <- predict(rf, test, type="class")

# 3. Confusion Matrix
rf$confusion
```

```
##      0   1 class.error
## 0 3515 357  0.09220041
## 1  684 718  0.48787447
```

```
accuracy_rf <- (rf$confusion[1,1]+rf$confusion[2,2])/(rf$confusion[1,1]+rf$confusion[1,2]+rf$confusion[
accuracy_rf
```

```
## [1] 0.8026166
```

```
pred_metrics("Random Forest Model",test$Churn.Value, pred_test)
```

```
## Random Forest Model
```

```
##    Accuracy Sensitivity Specificity
##   0.7969283   0.8384058   0.6455026
```

As we can see above, we first built the random forest model, predicted on train and test datasets, and computed the confusion matrix. One important note is to notice that this confusion matrix was extracted directly from the random forest model. The function randomForest in R already makes predictions based on the model created. We can again, predict on another datasets and compare results and create another confusion matrix to see that if necessary. That said, we can use the function pred_metrics() to calculate the accuracy, sensitivity and specificity from the predictions on the test dataset.

In this case, the accuracy of the random forest model was just slightly better than decision tree, and that can happen for different reasons. One reason is dataset size and variability, depending on those two factors, the different subsets of data and varibales selected for each tree can become very similar. Another possibility is that predicting only on the test set can result in different accuracies compared to predicting on the entire dataset. Using the test set predictions we can see metrics very similar to the decision tree model.

## Final Conclusions

Finally, once we are done with the complete analysis of a dataset, we need to compare the models to see which performs better, and possibly continue to tune it to achieve greater results. To make the comparison process easier, we can use the function pred_metrics previously created to calculate important classification metrics.

```
pred_metrics("Full Model",test$Churn.Value, predClass.full)
```

```
## Full Model
```

```
##    Accuracy Sensitivity Specificity
##   0.8179750   0.8577745   0.6832918
```

```
pred_metrics("Stepwise Regression Model",test$Churn.Value, predClass.step)
```

```
## Stepwise Regression Model
```

```
##    Accuracy Sensitivity Specificity
##   0.8174061   0.8581979   0.6806931
```

```
pred_metrics("Lasso Regression Model",test$Churn.Value, predClass.lasso)
```

```
## Lasso Regression Model
```

```
##    Accuracy Sensitivity Specificity
##   0.8174061   0.8576696   0.6815920
```

```
pred_metrics("Elastic Regression Model",test$Churn.Value, predClass.elnet)
```

```
## Elastic Regression Model
```

```
##    Accuracy Sensitivity Specificity
##   0.8179750   0.8572480   0.6842105
```

```
pred_metrics("KNN", y.test, pred.knn)
```

```
## KNN
```

```
##    Accuracy Sensitivity Specificity
##   0.7957907   0.8606811   0.6158798
```

```
pred_metrics("Decision Tree Model",test$Churn.Value, predicted_churn_score)
```

```
## Decision Tree Model
```

```
##    Accuracy Sensitivity Specificity
##   0.7980660   0.8111702   0.7204724
```

```
pred_metrics("Random Forest Model",test$Churn.Value, pred_test)
```

```
## Random Forest Model
```

```
##    Accuracy Sensitivity Specificity
##   0.7969283   0.8384058   0.6455026
```

From the classification metrics above, we can see that both the Lasso Regression and Elastic Regression models have slightly better metrics than the other models. Therefore, those could be the chosen ones to continue to tune and work with. It is important to some models such as decision trees when tuned, can out perform other logistic regression models. In conclusion, we can say that for this occasion, Lasso and Elastic Regression models outperformed the other ones.

### Implications of this Problem

The implications of analyzing datasets such as this one are very important for companies looking to retain important customers, not just telecommunications companies. The framework for the Linear Regression model shows that regression can be used to predict customer lifetime value using whatever data a company may have about a particular customer. The Logistic Regression models shows that companies could also produce classification models to predict the probability of a customer leaving. Both of these pieces of information

are valuable on their own, but combining them could add even more value to the company. For example, a company could use classification to determine which customers are most likely to churn, then sort those customers by lifetime value (found from the other model) to determine the high-priority customers to reach out to. This information helps the company focus its retention efforts and stay profitable because the most valuable customers are retained.