ST404 Assignment 3 Report- u2003245

Table of Contents

1- Abstract

Whether or not a customer 'Churns' is based on a number of factors, primarily:

- 'Complains'- a customer is much more likely to churn if they have had to complain
- 'CallFailure'- having failed calls makes a customer much more likely to churn
- 'Status'- non-active customers are more likely to churn
- 'FrequencyOfUse'- customers that make more calls are less likely to churn

Customers seem to not take into consideration how long they've had the phone contract when deciding to cancel it- with 'SubscriptionLength' having minimal impact on 'Churn'.

The final model performs highly when tested on the validation dataset however it may not generalise to the current day or other countries well due to technology playing a more prominent role in everyday life compared to when the data was collected.

2- Splitting the data

Initially I split the dataset into two sets- a training set which we will use to build our models and a validation set which will be used to test, evaluate and validate these models. For this we used 70% (2205) of the 3150 observations for the training set and the remaining 30% (945) used for the validation set; this was done as "Empirical studies show that the best results are obtained if we use 20-30% of the data for testing, and the remaining 70-80% of the data for training" (Gholamy, Kreinovich and Kosheleva: 2018)[1] and so I picked the lower end of this scale to ensure testing is more rigorous (as this means there is more data in the validation set). Splitting this data was done in a random fashion by creating a random vector consisting of 2205 random selected distinct integers from 1-3150 and using these indices for the rows selected by the training set (with the remainder being used for the validation set).

3- EDA

The next step in the process was conducting explanatory data analysis (EDA) in order to gain a good understanding of the data. Here we see how the variables are distributed and the correlation between them before finally assessing whether any of the (continuous explanatory) variables may require a transformation when it comes to building out models.

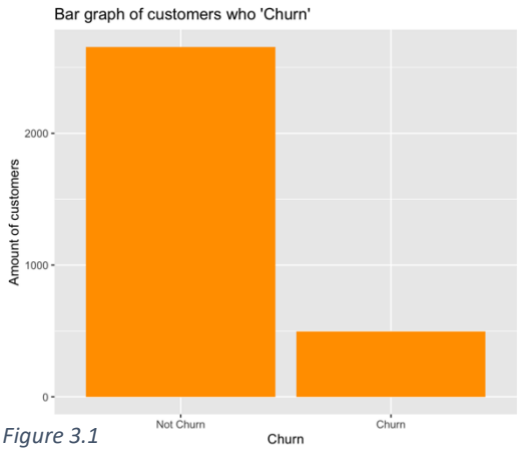

Figure 3.1

Looking at the distribution of the response variable- Churn- we can see from this bar graph (Figure 3.1) that the vast majority (84.3%) of customers do not churn which is good as it is generally consistent with what we would logically expect.

Most of the explanatory variables interactions with the response variable are what we would expect, for example customers being more likely to Churn when they have had to complain, however there are a few surprises, namely the interactions of 'Churn' with: 'SubscriptionLength', 'ChargeAmount" and 'AgeGroup'.

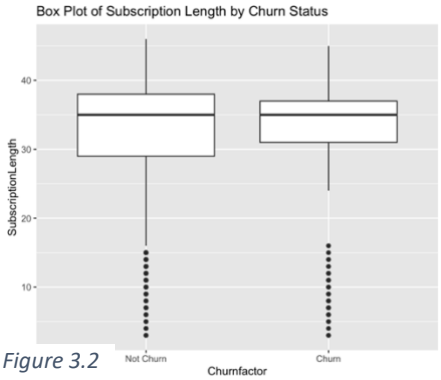

Figure 3.2

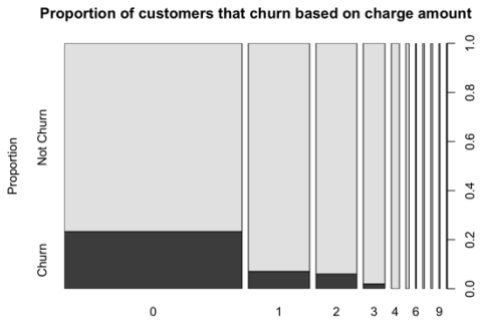

Figure 3.3

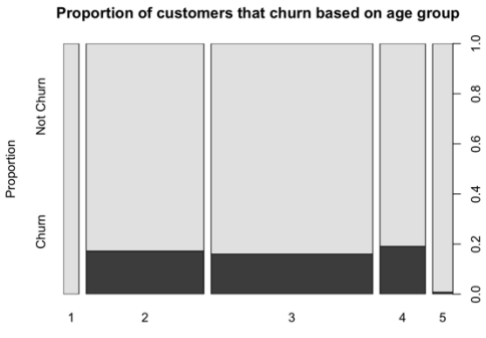

Figure 3.4

Looking at figure 3.2 we can see that the distribution of customer's subscription length is almost identical regardless on whether they churn or not. The boxplot shows that both groups have a median of around 35 months and whilst the interquartile range is larger for the 'not churn' group this is to be expected as there are much more customers (84.3% of the total) within this group. Intuitively, we would expect customers to be less likely to churn if they have had been with their provider for longer however this does not seem to be the case with customers seeming not to take this into account when deciding whether to 'churn'.

Charge amount is also something that would be anticipated to have a positive correlation with Churn- with a higher charge making a customer more likely to churn however as seen in figure 3.3 the opposite is actually true. This is most likely due to three key reasons: firstly customers that pay more would generally value their phone plans and so are less likely to switch, secondly the majority of customers are in the lower 'ChargeAmount' groups- so the amount of customers is not equally distributed- causing a bias in these lower charge groups and thirdly there is likely a large correlation between 'ChargeAmount' and variables such as 'FrequencyOfUse' which would typically have a lower churn rate as frequency increases and we will investigate this correlation further later.

Age group seems to be evenly distributed between groups 2-4 which (Keramati and Ardabili: 2011)[2] suggest is ages 15-60 and outside this range Churn likelihood is a lot less which is likely due to children (below 15) not being responsible for their contracts and rather their parents who are less likely to change this. The group over 60 are also much less likely to churn due to generally being more disinterested in technology so are less likely to change their phone contract.

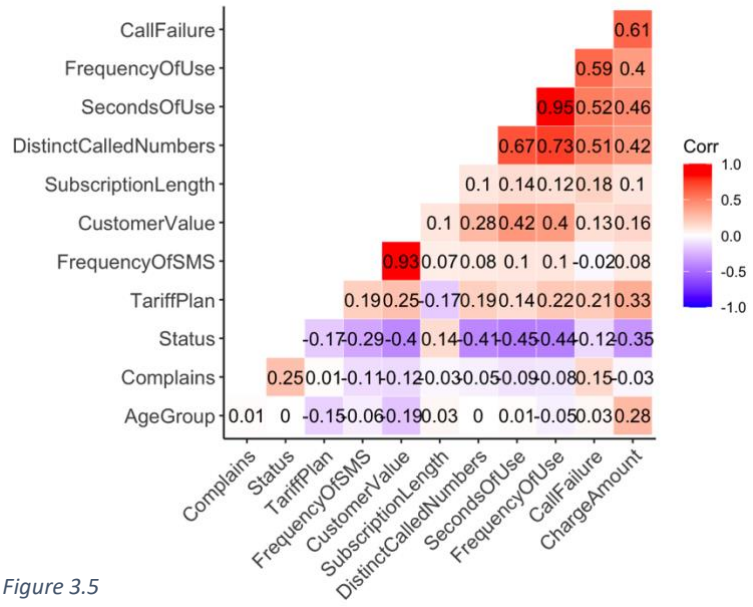

Figure 3.5

As previously mentioned, it is important to consider correlation between the explanatory variables as this can mean that a change of one variable causes an indirect effect in another. Looking at figure 3.5 we can see a correlation matrix of the explanatory variables and we can see that some variables are highly correlated, both positively and negatively. Many of these are to be expected such as a high correlation of 0.95 between 'SecondsOfUse' and 'FrequencyOfUse' however there are some

surprising correlations such as ‘SecondsOfUse’ and ‘Status’ having a correlation of -0.45 which is quite large. It is important to keep in mind which of these variables have high correlations when it comes to fitting our models to avoid issues caused by multicollinearity. Correlation of the response variable and the explanatory variables is also a very important consideration as this allows us to see which variables are most important in determining whether a customer churns. From Figure 3.6 we can see that ‘CustomerValue’ and ‘CallFailure’ have a very high correlation with ‘Churn’ making them extremely influential whilst ‘AgeGroup’, ‘SubscriptionLength’ and ‘CallFailure’ have almost zero correlation with Churn so are not as influential.

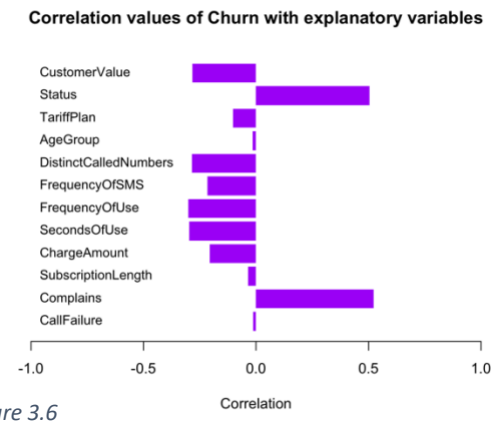

Figure 3.6

Another important consideration of our EDA is determining if any of the continuous explanatory variables require transformations when fitted in a model. A key example of this is ‘SecondsOfUse’. Figure 3.7 below is a histogram, boxplot, conditional histogram and conditional density plot (in this order) for ‘SecondsOfUse’ and figure 3.8 is the same but after taking a square root transformation. We can see that the frequency is much more normally distributed after this transformation which ensures the linearity assumption of the model is met so is ideal, when it comes to fitting our models. Furthermore it maintains the relationship with ‘Churn’ as can be seen by the boxplots being similar before and after the transformation was made. The conditional histograms and conditional density plots are much more normally distributed after the transformation too which is ideal as again it helps meet the linearity assumption.

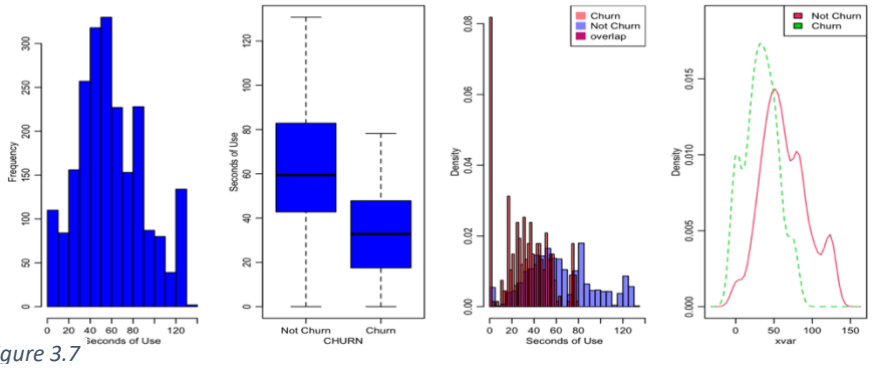

Figure 3.7

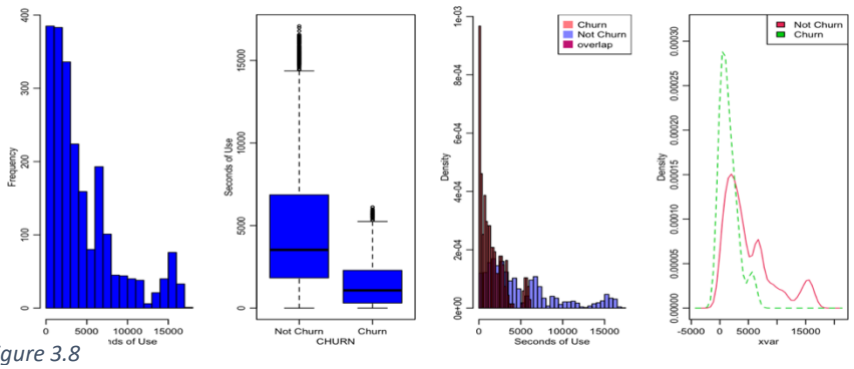

Figure 3.8

4- Creating models

4.1- Initial model

After concluding with the EDA, the next step was fitting an initial model which contains all of the variables in

some regard (i.e. they may have had a transformation applied to them). In our EDA we noticed that the variables: 'CallFailure', 'SubscriptionLength', SecondsOfUse', 'FrequencyOfUse', 'FrequencyOfSMS' and 'DistinctCalledNumbers' may all require transformations which were: log, squared, square root, square root, cube root and square root transformations, respectively. Out of all these transformations only the one applied to 'FrequencyOfSMS' was abundantly clear that the transformation needs to be applied and the original variable should be excluded from the initial model so for the rest we included both the original and transformed variable.

Since the response variable 'Churn' was a binary variable we used a GLM (generalised linear model) with a binomial family to best predict whether a customer will churn or not and we used 'Churn' as a factor variable with 2 levels (Churn and not Churn).

This gave us an initial model predicting 'Churn' from the explanatory variables:

```
model1 <- glm(Churnfactor~ CallFailure+log(CallFailure+0.5)+SubscriptionLength+(SubscriptionLength^2)+
                SecondsOfUse+sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(FrequencyOfUse)+
                I(FrequencyOfSMS^(1/3))+DistinctCalledNumbers+sqrt(DistinctCalledNumbers)+
                Complains+TariffPlan+Status+ChargeAmount+AgeGroup,family = "binomial")
```

From the parameter estimate table (using the summary function in R) we see some of what we would generally expect based on the findings from the EDA such as 'AgeGroup' having a large p-value meaning it is not very influential and 'Complains' having a very low p-value so it is. Surprisingly, not all the p-values match up to what we expected from the EDA- for example 'CallFailure' has a low p-value despite us finding it to have a low correlation with 'Churn' found in the EDA.

Due to having so many highly correlated variables as well as there being certain variables with more than one version (e.g. 'FrequencyOfUse' having both the original and a square root transformation in the model) there is an issue of multicollinearity which can be seen by several of the variables having a VIF of over 10- this is something we consider more later when fitting our final model.

4.2- Reduced model- automated approach

We used the LASSO (Least Absolute Shrinkage and Selection Operator) method as an automated approach to reduce the model. This method works by extracting the regression coefficients and then individually penalizing them, using a penalty term in a cost function, by tending some of the coefficients to zero. Then it works out the optimal values for each of these coefficients, ultimately getting rid of some of the variables, hence selecting only a subset of the original explanatory variables. This gave us the model:

```
model2 <- glm(Churnfactor~ CallFailure+log(CallFailure+0.5)+SubscriptionLength+(SubscriptionLength^2)+
                sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(FrequencyOfUse)+I(FrequencyOfSMS^(1/3))+
                DistinctCalledNumbers+sqrt(DistinctCalledNumbers)+Complains+TariffPlan+Status+ChargeAmount+
                AgeGroup,family = "binomial")
```

4.3- Reduced model- shrinkage approach

To reduce the model using a shrinkage approach we used stepwise selection using AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) (in both directions) and then removed the variables that both methods suggested removing. For both of the methods- in the backwards direction- they start with a full model then systematically remove them to get a model with the lowest AIC/BIC value which is preferred. (For the forwards direction they start with the null model and then systematically add the variables). Then this gave us 2 models- one for AIC and one for BIC- and so I created a model that removed the variables that both agreed upon removing. This gave us the model:

```
model3 <- glm(Churnfactor~ log(CallFailure+0.5)+SubscriptionLength+
                SecondsOfUse+sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(FrequencyOfUse)+
                I(FrequencyOfSMS^(1/3))+DistinctCalledNumbers+sqrt(DistinctCalledNumbers)+
                Complains+Status,family = "binomial")
```

5- Evaluating the models

In order to select a final model, we need to first evaluate all of the models and determine which is best. In all these methods of evaluating the models we used the validation dataset we initially created (as mentioned in Section 1). This is because the models have been fitted using the training dataset and so in order to see how well these models perform it is best to test them on unseen data which is precisely what the validation dataset is made from. Therefore, using the validation dataset, we can choose the best one out of our three candidate models ensuring it is more likely to work with new data rather than just being a perfect representation of the training data- meaning we overfitted the model according to that.

5.1- Potential profit evaluation

One way to evaluate each of the models is by using the cost matrix given in the assignment briefs. This can be used to work out the total potential profit for each customer by summing up each individuals customers potential profit- which are outlined in this table:

Profit matrix	Actually churns	Does not churn.
Predicted to Churn	0.6* Customer value - 70	Customer value - 70
Predicted not to Churn	0	Customer value

The initial model gave us a potential profit of $365197.5, the LASSO model $365335.1 and the step model $365136.3 for the customers in the validation data set. As we can see these values are all extremely alike, so more analysis is required to determine the best but since the step model is the ‘simplest’ (i.e. contains the least parameters) it seems to be preferrable in order to avoid overfitting.

5.2- ROC chart evaluation

Another method to evaluate these models is using ROC (Receiver Operating Characteristic) Curves. This is the true positive rate (on the y-axis) plotted against the false positive rate (on the x-axis). This plot is then evaluated using AUC (area under the curve), which simply works out the area under the ROC curve. A good classifier/model would have an ROC curve that 'hugs' the top right of the graph and hence you are looking for a large AUC. AUC is measured between 0.5, being equivalent to random guessing, and 1, which is a perfect classifier. As you can see in figure 5.2.1 all three of our models follow the trend we are hoping for- with them 'hugging' the top right corner. The AUC values for the models are as follows: initial model- 0.9208, LASSO model- 0.92, step model- 0.9208. Therefore, all the models have an extremely large AUC value which is ideal however it doesn't point towards one model as the best as they all perform very similarly.

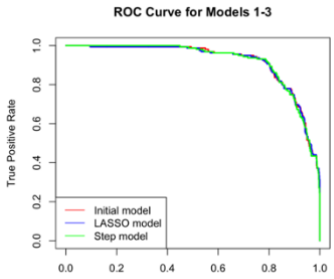

Figure 5.2.1

5.3- Fitting the final model

Since all the models performed similarly in my testing, I created a model which contains all parameters that are present in all of the models, giving us the 'intersection model':

```
model4 <- glm(Churnfactor~ log(CallFailure+0.5)+SubscriptionLength+
                    SecondsOfUse+sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(FrequencyOfUse)+
                    I(FrequencyOfSMS^(1/3))+DistinctCalledNumbers+sqrt(DistinctCalledNumbers)+
                    Complains+Status,family = "binomial")
```

Since multicollinearity was previously an issue, I investigated this further by plotting the VIFs of the explanatory variables which you can see in figure 5.3.1. Several of the variables have a VIF over the threshold of 10 which is indicated by the dotted blue line. These variables are: 'SecondsOfUse', 'sqrt(SecondsOfUse)', 'FrequencyOfUse', 'sqrt(FrequencyOfUse)', 'DistinctCalledNumbers' and 'sqrt(DistinctCalledNumbers)' so we can see that they are all variables with another version of the bariable present in the model so a high VIF and hence multicollinearity is expected. In order to remove these variables, I used the parameter estimates (from the summary function) and removed the least influential version of these three variables (after carefully interpreting the parameters). This gave us the final model:

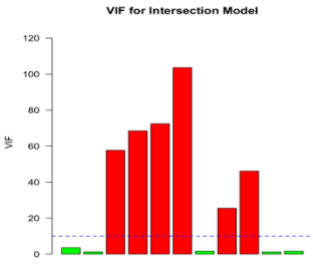

Figure 5.3.1

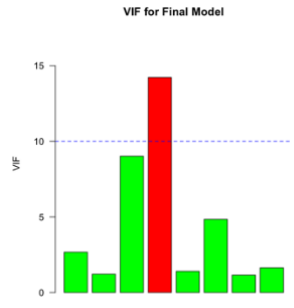

Figure 5.3.2

```
finalmodel <- glm(Churnfactor~ log(CallFailure+0.5)+SubscriptionLength+
                    sqrt(SecondsOfUse)+sqrt(FrequencyOfUse)+
                    I(FrequencyOfSMS^(1/3))+sqrt(DistinctCalledNumbers)+
                    Complains+Status,family = "binomial")
```

Checking multicollinearity of this final model we can see from figure 5.3.2 that only one variable has a VIF over 10 so this is a considerable improvement. We keep all these variables in as just a high VIF alone is not enough reason to remove a variable.

To evaluate the final model, we again checked the potential profit which came out to be $364563.30 which is extremely similar to the other models whilst being simpler so is preferred. We also again look at the AUC based on the ROC curve which is 92.05% which is only a 0.03% decrease from the previous best performing model and so is a very good model that fits the data well.

Section 6: Interpreting the final model

6.1- Interpreting the parameter estimates

The parameter estimates for our final model are as follows (to 5 decimal places):

Parameter	Intercept	log(Call Failure + 0.5)	Subscription Length	sqrt(Seconds OfUse)	sqrt(Frequency OfUse)	I(Frequency OfSMS^(1/3))	sqrt(Distinct CalledNumbers)	Complains	Status
Coefficient	0.42394	0.69976	-0.03903	0.00484	0.48261	-0.23718	0.00597	4.11293	1.78355

Interpretation of 'SubscriptionLength':

The continuous explanatory variable 'SubscriptionLength' has a parameter estimate of -0.03903. This means a one unit increase of 'SubscriptionLength' (i.e. a subscription that is one month longer) causes a decrease in the log odds of Churning by -0.03903 (holding all other variables constant). For example, if the log odds of churning are 0 there is a 50% probability of churning (using the formula: Probability of churning = exp(log odds) / (1 + exp(log odds))). So, in this example if we increase the subscription length by one month the probability of churning is 49.02441% causing a decrease of 0. 97559% in the probability of churning.

Interpretation of 'Complains':

The factor variable 'Complains' has a parameter estimate of 4.11293 meaning that a a customer having to complain increases the log odds of churning by 4.11293 (holding all other variables constant).

Therefore, for an example where no complaints have been made and the log odds of churning are zero there is a 50% probability of churning but if the customer complained the probability of churning will be 98.39035% causing an increase of 48.39035% in this example.

6.2- Illustrative example of the final model

We can use the final model to calculate the log odds ratio of churning and hence then the probability of churning. The variables we need for the illustration given in 6b of the brief are:

Variable	CallFailure	Complains	Subscription Length	SecondsOfUse	FrequencyOf Use	FrequencyOf SMS	DistinctCalledNumbers	Status
Value	10	yes	20	4000	80	70	10	Active

Our model has formula:

$$logodds = intercept + 0.699755583 \times \log(CallFailure + 0.5) - 0.039028663 \times SubscriptionLength + 0.004836548\sqrt{SecondsOfUse} - 0.48260696\sqrt{FrequencyOfUse} - 0.237179669\sqrt[3]{FrequencyOfSMS} + 0.005965162\sqrt{DistinctCalledNumbers} + 4.112926465 \times 1\{Complains = yes\} + 1.783546295 \times 1\{Status = non - active\}$$

So in our case we have:

$$logodds = 0.423940290 + 0.699755583 \times log(10 + 0.5) + -0.039028663 \times 20 + 0.004836548 \times \sqrt{4000} + -0.482606961 \times \sqrt{80} - 0.237179669 \times \sqrt[3]{70} + 0.005965162 \times \sqrt{10} + 4.112926465 \times 1 + 1.783546295 \times 0$$

$$\Longrightarrow logodds = 0.4323821$$

Using the equation: $probability\ of\ churning = \frac{\exp(\text{logodds})}{1+\exp(logodds)}$

We get: the probability of churning = 60.64424% which is higher than 50%, so the model predicts that this customer will churn.

Section 7: Further validation of the final model

7.1- Precision-Recall curve

A further way to validate the final model is by plotting a precision- recall curve. This plots precision on the y-axis against recall on the x-axis. Precision is the ratio of true positives to the total number of predicted positives (true positives + false positives) and recall is the ratio of true positives to the total number of actual positives (true positives + false negatives). So we are looking for a model that has both high precision and recall- i.e. it 'hugs' the top right corner of the graph. From figure 7.1.1 we see this, further validating that the final model fits the data well.

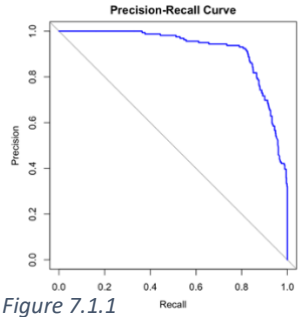

Figure 7.1.1

7.2- Calibration plot

Another plot we can use to evaluate the model is the calibration plot. This plots the observed outcome rate against the mean predicted probabilities so we are looking for a plot that follows the y=x line (the dashed line in figure 7.2.1). As we can see the data follows this line and whilst it is not perfect the line of best fit is very close to the y=x line indicating we have a good predictive model.

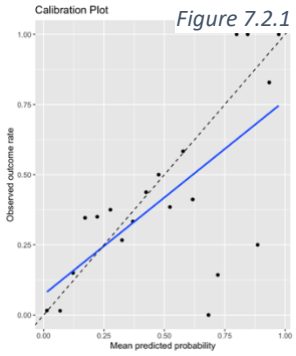

Figure 7.2.1

Section 8: Limitations of the model

One limitation of the model is that all the data collected was done in a span in 9 months and the churn was measured 3 months later so it may not include results outside this time period that could affect the outcome (e.g. a customer complaining one day after these 9 months). As well as this it may be hard to generalise this model to the current day as the data was collected in the 20th century and mobile phones have become much more prominent in everyday life since then meaning factors affecting 'churn' rate may be very different today. Given more time I would have liked to perform more analysis in the EDA stages particularly with testing more transformations such as Box-Cox and Turkey transformations. Furthermore, I would have liked to test interactions between the explanatory variables to see the effect of that on the response variable.

Section 9: Word Count

Word count: 2842 words

Section 10: References

[1] https://scholarworks.utep.edu/cs_techrep/1209/

[2] Keramati , Abbas, and Seyed M.S. Ardabili. 2011. “Churn analysis for an Iranian mobile operator.” Telecommunications Policy 35 (4): 344-356. doi:https://doi.org/10.1016/j.telpol.2011.02.009.

Section 11: Appendix

11.0- Preliminaries

First we load in the data and look at a summary of it.

```
load("TeleChurn.rdata")
summary(TeleChurn)
head(TeleChurn)
```

Then we install all the packages we need

```
library(car)
library(olsrr)
library(VIM)
library(ggplot2)
library(relaimpo)
library(caret)
library(MASS)
library(gglasso)
library(glmnet)
library(Stat2Data)
library(pROC)
library(rgl)
library(reshape)
library(gclus)
library(sm)
library(ggpubr)
library(plyr)
library(leaps)
library(calibrate)
library(yardstick)
library(Information)
library(ggcorrplot)
library(corrplot)
library(dplyr)
```

Then we set all the factor variables as factors. For example ‘Complains’:

```
TeleChurn$Complains <- as.factor(TeleChurn$Complains)
```

11.1- Splitting the data

```
set.seed(123)
n_train <- 0.7 * nrow(TeleChurn) #sets 70% of the data as training and hen
ce 30% for validation
```

```
train_indices <- sample(3150, size = n_train, replace = FALSE)
train <- TeleChurn[train_indices, ]
valid <- TeleChurn[-train_indices, ]
attach(train)
```

11.2- EDA

First we look at the response variable

```
ggplot(TeleChurn, aes(x = Churnfactor)) +
  geom_bar(fill="orange") +
  ggtitle("Bar graph of customers who 'Churn'") +
  labs(y= "Amount of customers", x= "Churn")
table(TeleChurn$Churn)
```

Then we look at all the continuous explanatory variables. For example CallFailure:

```
ggplot(train, aes(x = Churnfactor, y = CallFailure)) +
  geom_boxplot()
hist(CallFailure)
```

Then we look at the factor explanatory variables. For example Complains:

```
table(Complains)
plot(Complains, Churnfactor)
```

Then we look at the correlation matrix of the explanatory variables.

```
corrdata<- apply(train, 2, as.numeric)
corrdata <- corrdata[, -ncol(corrdata)]
corrdata1 <- corrdata
corrdata <- corrdata[, -ncol(corrdata)]
corr_matrix <- cor(corrdata)
ggcorrplot(corr_matrix,
           hc.order = TRUE,
           type = "lower",
           outline.color = "white",
           colors = c("blue", "white", "red"),
           ggtheme = ggplot2::theme_classic(),
           lab = TRUE)
```

Then we look at the correlation between the response and explanatory variables.

```
corr_matrix1 <- cor(corrdata1)
corr_matrix2 <- corr_matrix1[,ncol(corr_matrix1)]
corr_matrix2 <- corr_matrix2[-13]
bar_heights <- barplot(corr_matrix2,
                       horiz = TRUE,
                       main = "Correlation values of Churn with explanator
y variables",
                       xlab = "Correlation",
                       xlim = c(-1, 1),
                       col = "purple",
                       border = NA,
                       yaxt = "n")
text(x = -1,
```

```
     y = bar_heights,
     labels = names(corr_matrix2),
     pos = 4,
     col = "black",
     cex = 0.9,
```

Then we plot the conditional densities using ggplot

```
TeleChurn.long <- melt(TeleChurn[,c(1:11,14)], id="Churnfactor")
ggplot(aes(x=value, group=Churnfactor, col=factor(Churnfactor)), data=Tele
Churn.long) +
  geom_density() + facet_wrap(~ variable, scales="free")
```

Then using the sm package. For example CallFailure:

```
sm.density.compare(CallFailure, Churn)
title(main="CallFailure against churn")
```

Then we look at a barchart of churn vs not churn.

```
barplot(prop.table(summary(Churnfactor)),col="red",main="Bar Chart of Chur
n Test",
        ylab="Proportion of units in study",xlab="Result of Test for Churn
", ylim=c(0,1),
        axis.lty=1)
```

Then we look at conditional density plots for the continuous explanatory variables. For example CallFailure:

```
ggplot(TeleChurn, aes(x = Complains, fill = Churnfactor)) +
  geom_density(alpha = 0.5) +
  ggtitle("Conditional Density Plot of complains by Churn Status")
```

Then we look at the empirical logits.

```
# function to calculate bins
cut.equal <- function(x, N) {
  breaks <- quantile(x, seq(from=0, to=1, length=N+1))
  cut(x, unique(breaks))
}

## Empirical logits
emplogit <- function(x, y, ngroups, ...) {
  xgroup <- cut.equal(x, ngroups)
  xmean <- tapply(x, INDEX=xgroup, FUN=mean)
  ylogit <- tapply(y, INDEX=xgroup, FUN = function(y) {
    log(sum(y) + 0.5) - log(sum(1-y) + 0.5)
  })
  plot(xmean, ylogit, ylab='empirical logit',...)
}

par(mfrow=c(3,3), mar=c(4.1, 4.1, 2.1, 2.1))
nam <- colnames(TeleChurn)
for (k in c(1,3,5,6,7,8)){
  emplogit(TeleChurn[,k], as.numeric(TeleChurn[,13]), ngroups=15, main=nam
```

```
[k], type="b", pch=16)
}
par(mfrow=c(1,1))
barplot(prop.table(summary(Churnfactor)),col="red",main="Bar Chart of chur
n Test",
        ylab="Proportion of people who churn",xlab="Result of Test for chu
rn", ylim=c(0,1),
        axis.lty=1)

# Method 2 for empirical logit:
myemplogit <- function(yvar=y,xvar=x,maxbins=10,sc=1,line=TRUE,...){
  breaks  <<- unique(quantile(xvar, probs=0:maxbins/maxbins))
  levs  <<- (cut(xvar, breaks, include.lowest=FALSE))
  num <<- as.numeric(levs)
  c.tab <- count(num,'levs')
  c.tab$levs <- factor(c.tab$levs, levels = levels(addNA(c.tab$levs)), lab
els = c(levels(c.tab$levs),

paste("[",min(xvar),"]",sep="")), exclude = NULL)
  c.tab <- c.tab[c(nrow(c.tab),1:nrow(c.tab)-1),]
  sc <- (max(c.tab$freq)/min(c.tab$freq)/sc)^2
  zcex <<- sqrt(c.tab$freq/pi)/sc
  print(c.tab);print(zcex);print(sc)
  emplogitplot1(yvar~xvar,breaks=breaks,cex=1,showline=line,...)
}
```

And view these, e.g. for CallFailure:

```
myemplogit(Churnfactor,CallFailure,30,sc=30,xlab="Call Failures")
```

Then we create a function for the conditional histograms.

```
#Define Colours
myblue <- rgb(0, 0, 255, max = 255, alpha = 125, names = "blue50")
myred <- rgb(255, 0, 0, max = 255, alpha = 125, names = "red50")
myboth <- rgb(191, 66, 130, max = 255, alpha = 255, names = "rednblue50")
#Function for conditional histograms
conHist <- function(yvar=y,xvar=x,br=18,yvar1=0,yvar2=1,leg=FALSE,xxlab=de
parse(substitute(xvar)),cex.l=1,cex.a=1,title="",cex.m=1,...){
  h1 <-  graphics::hist(xvar[yvar==yvar1],plot=FALSE,breaks=br)
  h2 <-  graphics::hist(xvar[yvar==yvar2],plot=FALSE,breaks=br)
  mxy <- max(max(h1$density),max(h2$density))
  mxx <- max(c(max(h1$breaks),max(h2$breaks)))
  mnx <- min(c(min(h1$breaks),min(h2$breaks)))
  graphics::hist(xvar[yvar==yvar1],xlab=xxlab,prob=TRUE,col=myblue,
                 breaks=br,
                 ylim=c(0,mxy),xlim=c(mnx,mxx),cex.lab=cex.l,cex.axis=cex.
a,cex.main=cex.m,main=title)
  graphics::hist(xvar[yvar==yvar2],xlab="",prob=TRUE,col=myred,
                 breaks=br,main="",
                 ylim=c(0,my),xlim=c(0,mx),add=TRUE)
  if (leg){
    legend('topright',legend=c(yvar2,yvar1,"overlap"),col=c(myred,myblue,m
yboth),pch=15,pt.cex=2)
```

```
  }
}
```

Then view these, e.g. for CallFailure:

```
conHist(Churnfactor, CallFailure,leg=TRUE,yvar1="Not Churn",yvar2="Churn",
cex.l=1.5,cex.a=1.5,title="Conditional Histogram Call Failure",cex.m=1)
```

Then we put these plots together

```
plotall <- function(yvar=y,xvar=x,br=br,yvar1=0,yvar2=1,leg=TRUE,maxbins=1
0,sc=1,line=TRUE,xxlab=deparse(substitute(xvar)),yylab="y",Title=xxlab,...
){
  par(mfrow=c(1,4),mgp=c(1.7,0.7,0),mar=c(3.5,3.5,1,1),oma=c(1,1,2.5,0))
  hist(xvar,col="blue",main="",xlab=xxlab)
  boxplot(xvar~yvar,xlab = yylab,ylab = xxlab, col="blue")
  conHist(yvar,xvar,leg=leg,yvar1=yvar1,yvar2=yvar2,xxlab=xxlab,br=br)
  sm.density.compare(xvar,yvar)
  title(main="")
  colfill<-c(2:(2+length(levels(yvar))))
  if(leg){
    legend("topright", levels(yvar), fill=colfill)
  }
  myemplogit(as.numeric(yvar)-1,xvar,maxbins,sc=sc,xlab=xxlab)
  emplogit(xvar, as.numeric(yvar)-1, ngroups=15, main="", type="b", pch=16
)
  mtext(Title, side = 3, line = 0, outer = TRUE,cex=1.8)
}
```

This allows us to investigate transformations so we evaulate these. e.g. for CallFailure:

```
plotall(Churnfactor,CallFailure,br=30,maxbins=30,yvar1="Not Churn",yvar2="
Churn",leg=TRUE,sc=30, yylab = "CHURN",xxlab="Call Failures")
plotall(Churnfactor,sqrt(CallFailure),br=30,maxbins=30,yvar1="Not Churn",y
var2="Churn",leg=TRUE,sc=30, yylab = "CHURN",xxlab="Call Failures")
plotall(Churnfactor,log(CallFailure),br=30,maxbins=30,yvar1="Not Churn",yv
ar2="Churn",leg=TRUE,sc=30, yylab = "CHURN",xxlab="Call Failures")
# here log looks best but also include original
```

11.3- Fitting models

Fitting the initial model and performing checks on it:

```
# initial model:
model1 <- glm(Churnfactor~ CallFailure+log(CallFailure+0.5)+SubscriptionLe
ngth+(SubscriptionLength^2)+
                SecondsOfUse+sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(Freque
ncyOfUse)+
                I(FrequencyOfSMS^(1/3))+DistinctCalledNumbers+sqrt(Distinc
tCalledNumbers)+
                Complains+TariffPlan+Status+ChargeAmount+AgeGroup,family =
"binomial")

# would be nice to do a turkey transformation test for subscription length
but don't have time
```

```
# brief evaluation of initial model:
summary(model1)
anova(model1)

# multicollinearity check for initial model:
vif(model1)
# large amount of multicollinearity for variables: SecondsOfUse, Frequency
OfUSe and DistinctCalledNumbers
```

Then create the LASSO model:

```
# Automated approach to reduce the model- LASSO:
# Convert the data to a matrix format
X <- model.matrix(model1)[,-1]

# Fit a Lasso model using cross-validation
cv_model <- cv.glmnet(X, Churnfactor, family = "binomial", alpha = 1)

# Get the optimal value of lambda
lambda_opt <- cv_model$lambda.min

# Fit a Lasso model using the optimal lambda
lasso_model <- glmnet(X, Churnfactor, family = "binomial", alpha = 1, lamb
da = lambda_opt)

# Get the coefficients of the Lasso model
lasso_coef <- coef(lasso_model)
print(lasso_coef)

# This gives us our second model:

model2 <- glm(Churnfactor~ CallFailure+log(CallFailure+0.5)+SubscriptionLe
ngth+(SubscriptionLength^2)+
                sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(FrequencyOfUse)+I(F
requencyOfSMS^(1/3))+
                DistinctCalledNumbers+sqrt(DistinctCalledNumbers)+Complain
s+TariffPlan+Status+ChargeAmount+
                AgeGroup,family = "binomial")
```

Then create the step model:

```
# Now use shrinkage approach to reduce the model- using AIC and BIC
model_aic <- stepAIC(model1, direction = "both", trace = FALSE, k = log(nr
ow(train)))
selected_aic <- names(coef(model_aic)[-1][which(coef(model_aic)[-1] != 0)]
)

# Perform stepwise selection with BIC
model_bic <- stepAIC(model1, direction = "both", trace = FALSE, k = log(nr
ow(train)), AIC = FALSE)
selected_bic <- names(coef(model_bic)[-1][which(coef(model_bic)[-1] != 0)]
)
```

```
# Combine selected variables
selected_vars <- unique(c(selected_aic, selected_bic))
selected_vars

# Build final model with selected variables

model3 <- glm(Churnfactor~ log(CallFailure+0.5)+SubscriptionLength+
                SecondsOfUse+sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(Freque
ncyOfUse)+
                I(FrequencyOfSMS^(1/3))+DistinctCalledNumbers+sqrt(Distinc
tCalledNumbers)+
                Complains+Status,family = "binomial")
```

11.4- Evaluating the models

Then we perform interim checks of these 3 models:

```
# Model1:
plot(model1)
vif(model1)
# Model2:
plot(model2)
vif(model2)
# Model3:
plot(model3)
vif(model3)
```

Then we create the profit function and evaluate the models using this.

```
# Profit function
evaluate_profit <- function(model, data) {
  prob_churn <- predict(model, newdata = data, type = "response")
  predicted_churn <- ifelse(prob_churn >= 0.5, 1, 0)


  profit <- ifelse(predicted_churn == 1 & data$Churn == 1, 0.6 * data$Cust
omerValue - 70,
                   ifelse(predicted_churn == 0 & data$Churn == 0, data$Cus
tomerValue - 70,
                          ifelse(predicted_churn == 0 & data$Churn == 1, 0
,
                                 ifelse(predicted_churn == 1 & data$Churn
== 0, data$CustomerValue, 0))))

  churnchurn <- sum(predicted_churn == 1 & data$Churn == 1)
  predchurnaccnot <- sum(predicted_churn == 1 & data$Churn == 0)
  prednotaccchurn <- sum(predicted_churn == 0 & data$Churn == 1)
  notnot <- sum(predicted_churn == 0 & data$Churn == 0)

# Calculate total profit
total_profit <- sum(profit)

# Return total profit
print(churnchurn)
print(predchurnaccnot)
```

```
print(prednotaccchurn)
print(notnot)
return(total_profit)

}

# now test for our models:
evaluate_profit(model1, valid)
evaluate_profit(model2, valid)
evaluate_profit(model3, valid)
# model2 has highest profit but very small difference
```

Then we look at ROC for the models and evaluate them using AUC:

```
# Now look at ROC curves:
roc_model1 <- roc(valid$Churnfactor, predict(model1, newdata = valid, type
= "response"))
roc_model2 <- roc(valid$Churnfactor, predict(model2, newdata = valid, type
= "response"))
roc_model3 <- roc(valid$Churnfactor, predict(model3, newdata = valid, type
= "response"))
# Create an empty plot with the appropriate axes labels and title
plot(0, 0, type = "n", xlim = c(0, 1), ylim = c(0, 1),
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curve for candidate models")

# ROC curve for model1 in red
lines(roc_model1, col = "red")

# ROC curve for model2 in blue
lines(roc_model2, col = "blue")

# ROC curve for model3 in green
lines(roc_model3, col = "green")

# Add a legend to the plot
legend("bottomleft", legend = c("Initial model", "LASSO model", "Step mode
l"),
       col = c("red", "blue", "green"), lty = 1)

# For model1:
auc_model1 <- auc(roc_model1)
auc_model1
# For model2:
auc_model2 <- auc(roc_model2)
auc_model2
# For model3:
auc_model3 <- auc(roc_model3)
auc_model3
# all 3 models have extrmely similar AUC
```

Then we create an intersection model of these 3 and check multicollinearity.

```
model4 <- glm(Churnfactor~ log(CallFailure+0.5)+SubscriptionLength+
                      SecondsOfUse+sqrt(SecondsOfUse)+FrequencyOfUse+sqrt(F
requencyOfUse)+
                      I(FrequencyOfSMS^(1/3))+DistinctCalledNumbers+sqrt(Di
stinctCalledNumbers)+
                      Complains+Status,family = "binomial")


# Multicollinearity was previously an issue so lets investigate this furth
er
vif(model4)
plot(vif(model4))
barplot(vif(model4))

vif_values4 <- vif(model4)
vif_threshold <- 10
barplot(vif_values4,
        col = ifelse(vif_values4 > vif_threshold, "red", "green"),
        ylim = c(0, max(vif_values4)*1.2))
abline(h = vif_threshold, col = "blue", lty = 2)
title(main = "VIF for Intersection Model")
xlabel <- "Variable"
ylabel <- "VIF"
title(xlab = xlabel, ylab = ylabel)

barplot(vif(model4), ylim = c(0, max(vif_values4)*1.2),
        ylab = "VIF",
        xlab= "Variables",
        main = "VIF for Intersection Model",
        col = ifelse(vif_values4 > 10, "red", "green"), las = 2, names.arg
=F)
abline(h = 10, lty = 2, col = "blue")
# From this we can see that all the variables with more than one version p
resent have a high VIF
# So lets see what the most important level is
coef(model4)
```

Then create the final model and evaluate it:

```
finalmodel <- glm(Churnfactor~ log(CallFailure+0.5)+SubscriptionLength+
                    sqrt(SecondsOfUse)+sqrt(FrequencyOfUse)+
                    I(FrequencyOfSMS^(1/3))+sqrt(DistinctCalledNumbers)+
                    Complains+Status,family = "binomial")

finalvif <- vif(finalmodel)
barplot(vif(finalmodel), ylim = c(0, max(finalvif)*1.2),
        ylab = "VIF",
        xlab= "Variables",
        main = "VIF for Final Model",
        col = ifelse(finalvif > 10, "red", "green"), las = 2, names.arg=F)
abline(h = 10, lty = 2, col = "blue")
roc_finalmodel <- roc(valid$Churnfactor, predict(finalmodel, newdata = val
id, type = "response"))
auc(roc_finalmodel)
```

```
# Still an extremely high AUC with only a 0.03% decrease from the previous
ly best performing model
evaluate_profit(finalmodel, valid)
# The predicted profit is also extremely similar to the previous models
# therefore we will take this as our final model
```

11.5- Interpreting the final model

Look at the parameter estimates for the final model.

```
coef(finalmodel)
```

Then we calculate the logodds and probability for the example.

```
logodds <- 0.423940290 + 0.699755583*log(10+0.5) + -0.039028663*20 + 0.004
836548*sqrt(4000) + -0.482606961*sqrt(80) + -0.237179669*(70^(1/3)) + 0.00
5965162*sqrt(10) + 4.112926465 + 1.783546295*0
logodds
prob <- exp(logodds)/(1+exp(logodds))
prob
```

11.6- Validating the final model

Precision- recall plot:

```
# Predict on validation set
pred <- predict(finalmodel, type = "response", newdata = valid)
# Create ROC object
pr <- roc(valid$Churnfactor, pred, plot = FALSE)
# Plot precision-recall curve
plot(pr, col = "blue", main = "Precision-Recall Curve", xlab= "Recall", yl
ab= "Precision", xlim=c(0,1), ylim=c(0,1))
# this is ideal
```

Calibration plot:

```
# predict on the validation set
valid$pred <- predict(finalmodel, newdata = valid, type = "response")
# create bins for predictions
bin_valid <- cut(valid$pred, breaks = seq(0, 1, by = 0.05))
# calculate mean predicted probability and observed outcome for each bin
calib_data <- aggregate(cbind(pred = pred, Churnfactor = (as.numeric(Churn
factor)-1)) ~ bin_valid, valid,
                        FUN = function(x) c(mean(x), sum(x)))
# create calibration plot
ggplot(calib_data, aes(x = pred[, 1], y = Churnfactor[, 1])) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, fullrange = TRUE) +
  labs(x = "Mean predicted probability", y = "Observed outcome rate")+
  geom_abline(linetype = "dashed") +
  ggtitle("Calibration Plot")
```