edX Capstone Telco Churn Analysis

Matt Rowse

28/08/2020

Project Overview

This report has been prepared for the edX HarvardX: PH125.9x capstone requirement and the data is available from Kaggle here.

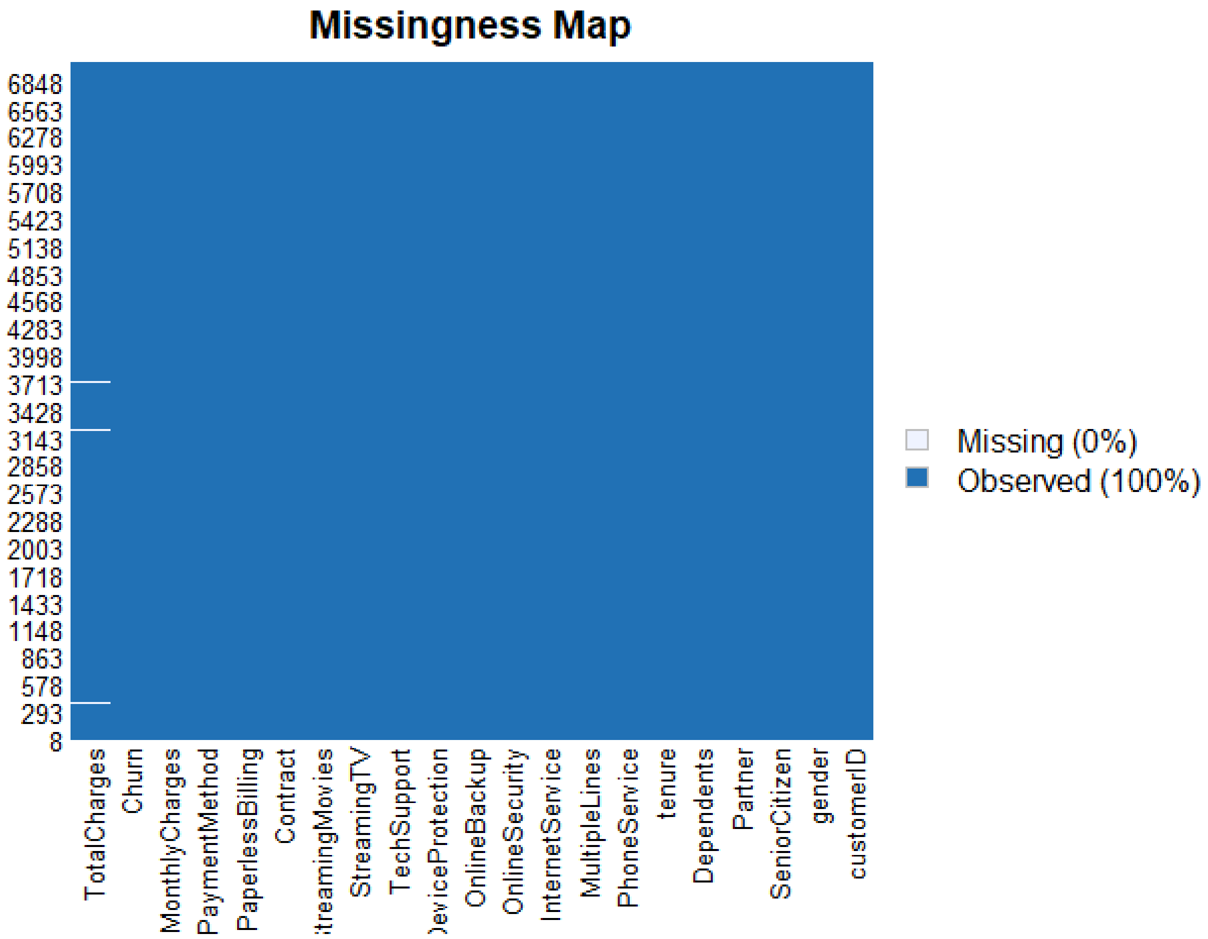
The data consists of 7043 unique customer values, with 21 columns and an overall target “Churn” column.

My objective in this project is to initially analyse and provide insights from the data and ultimately use this to train a Churn prediction model which could be used for busines purposes.

```
# Perform initial checks of the data
str(data)
```

```
## 'data.frame':    7043 obs. of  21 variables:
##  $ customerID      : Factor w/ 7043 levels "0002-ORFBO","0003-MKNFE",..
##  $ gender          : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 1 2
##  $ SeniorCitizen   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Partner         : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 2 1
##  $ Dependents      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 2
##  $ tenure          : int  1 34 2 45 2 8 22 10 28 62 ...
##  $ PhoneService    : Factor w/ 2 levels "No","Yes": 1 2 2 1 2 2 2 1 2 2
##  $ MultipleLines   : Factor w/ 3 levels "No","No phone service",..: 2 1
##  $ InternetService : Factor w/ 3 levels "DSL","Fiber optic",..: 1 1 1 1
##  $ OnlineSecurity  : Factor w/ 3 levels "No","No internet service",..:
##  $ OnlineBackup    : Factor w/ 3 levels "No","No internet service",..:
##  $ DeviceProtection: Factor w/ 3 levels "No","No internet service",..:
##  $ TechSupport     : Factor w/ 3 levels "No","No internet service",..:
##  $ StreamingTV     : Factor w/ 3 levels "No","No internet service",..:
##  $ StreamingMovies : Factor w/ 3 levels "No","No internet service",..:
##  $ Contract        : Factor w/ 3 levels "Month-to-month",..: 1 2 1 2 1
##  $ PaperlessBilling: Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 2 1
##  $ PaymentMethod   : Factor w/ 4 levels "Bank transfer (automatic)",..:
##  $ MonthlyCharges  : num  29.9 57 53.9 42.3 70.7 ...
##  $ TotalCharges    : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ TotalCharges     : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ Churn            : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 2 1 1 2 1
```

```
missmap(data)
```

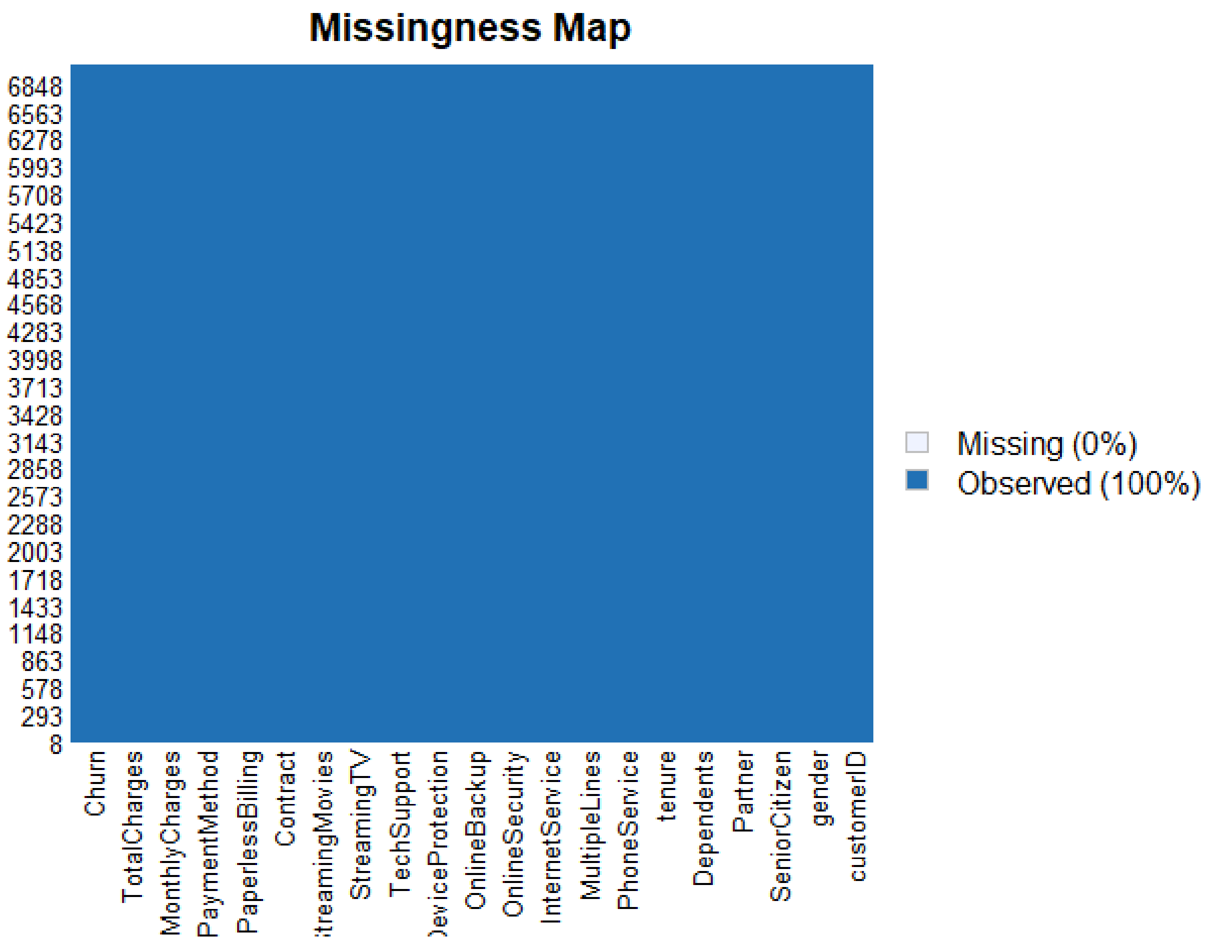


Wrangling

Note there are a small number of missing values in the 'TotalCharges' observations. My decision here is to predict a mean value for this missing observation and use all available data, rather than ignore this information completely.

It appears the SeniorCitizen observations have unique values of either 0 or 1 and are actually a factor and these formats will be coerced.

```
# Replace the na total charges with the mean
data <- data  %>%
  mutate(TotalCharges = if_else(is.na(TotalCharges) == TRUE,
  mean(TotalCharges, na.rm = TRUE), TotalCharges))
missmap(data)
```



```
# View the unique SeniorCitizen values and convert to factor
unique(data$SeniorCitizen)
```

```
## [1] 0 1
```

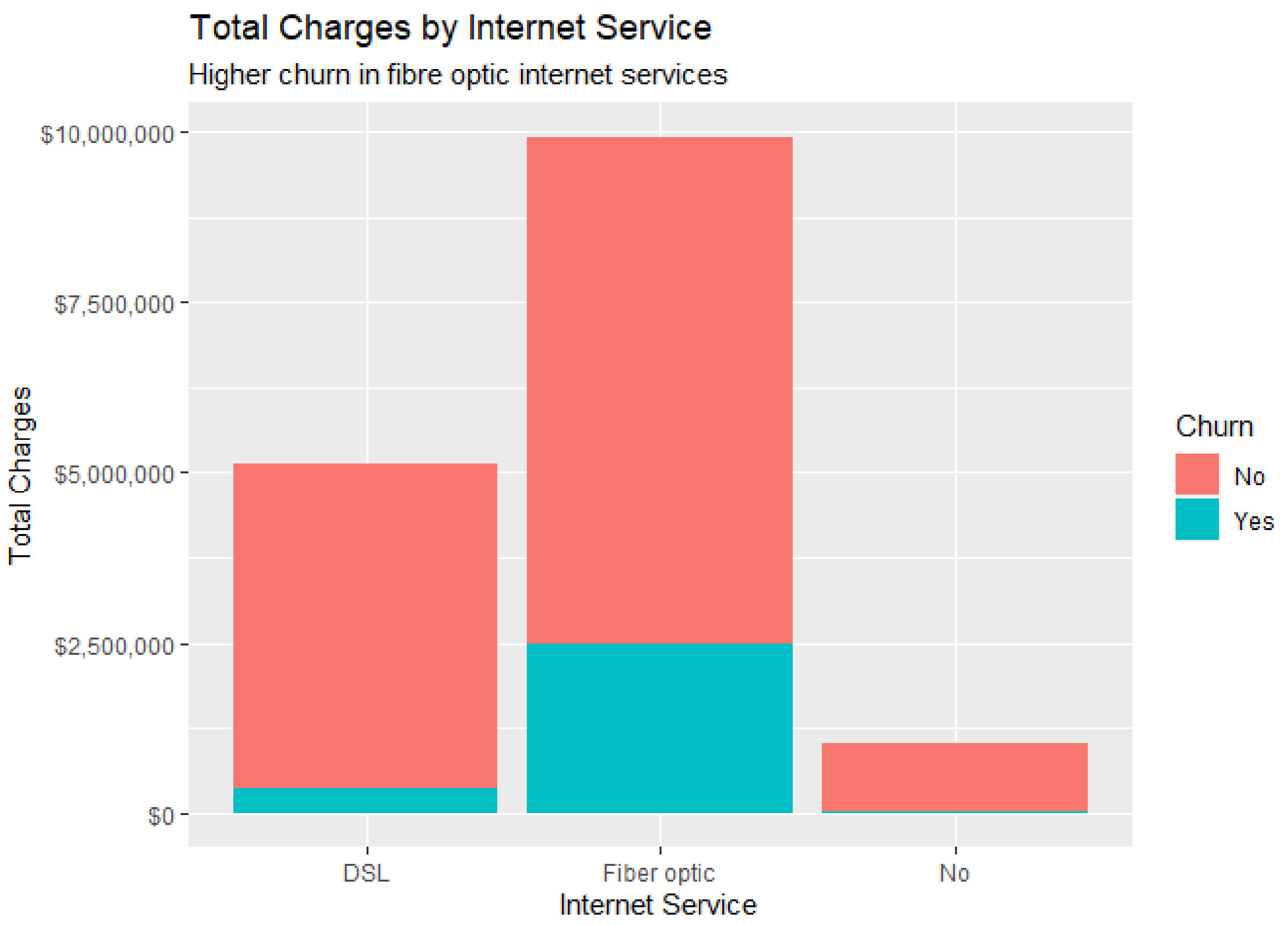
```
# Coerce
data <- data %>%
  mutate(SeniorCitizen = as.factor(if_else(SeniorCitizen == 1, "Yes", "No"
data <- data %>%
  mutate(tenure = as.integer(tenure))
```

Expolration

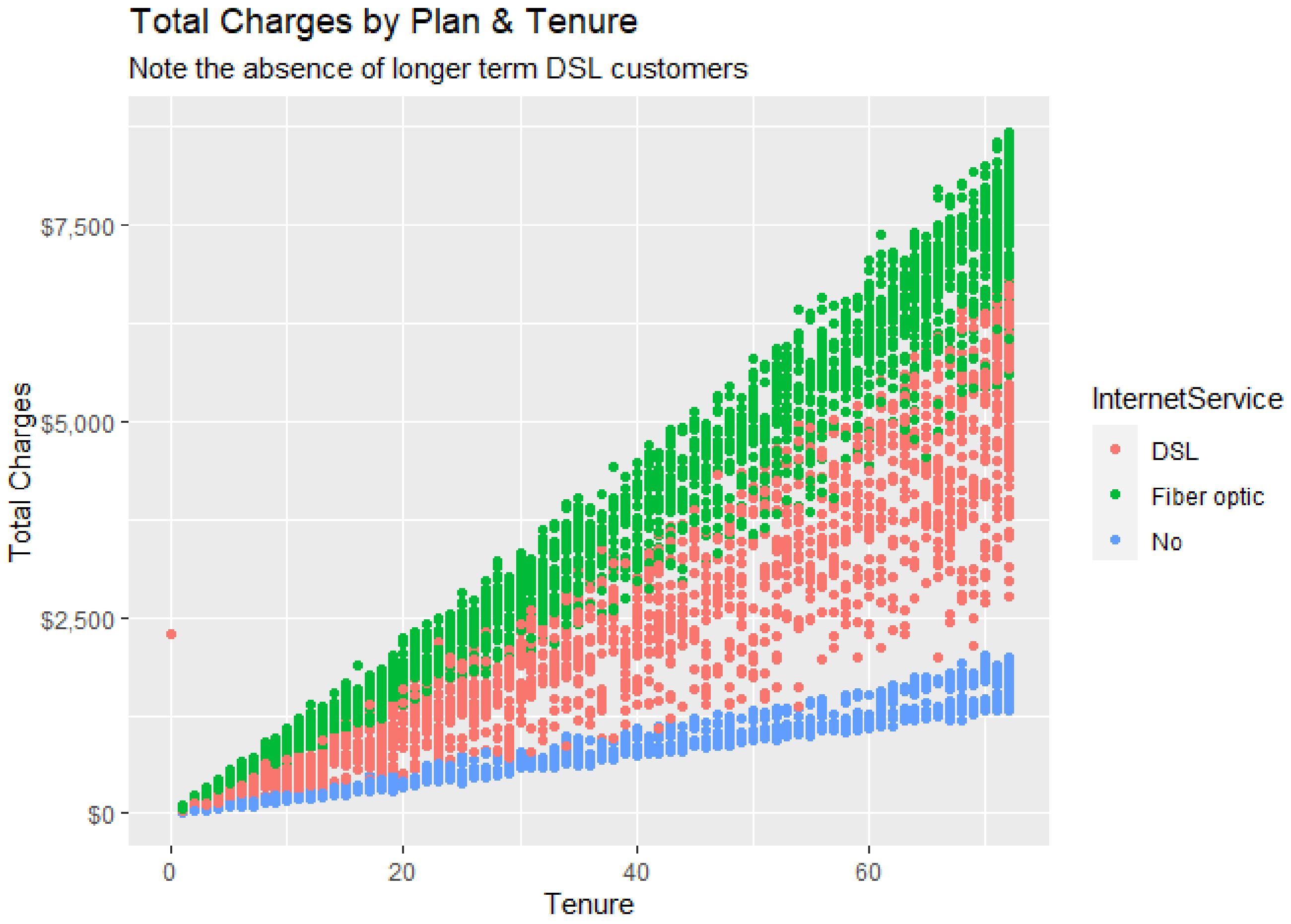
Now that the dataset is complete let’s gather some some insights through exploratory data analysis (EDA).

The aim here is to understand key insights for why customers may be churning to identify trends, which can provide a basis for future strategy.

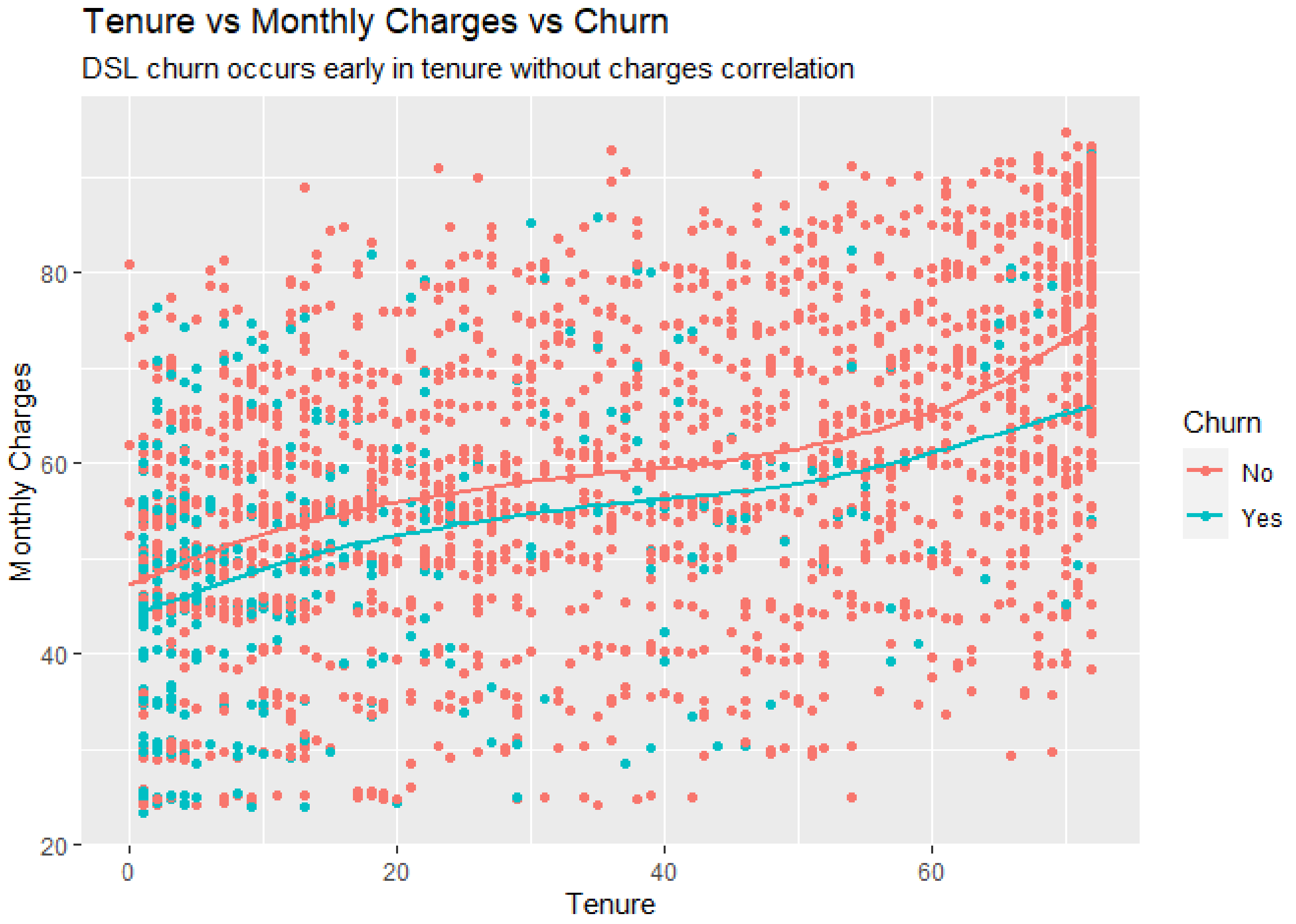
```
# Churn vs internet service
data %>% ggplot(aes(InternetService, TotalCharges, fill = Churn))+
  geom_col()+
  ggtitle("Total Charges by Internet Service")+
  labs(x="Internet Service", y="Total Charges",
  subtitle = "Higher churn in fibre optic internet services")+
  scale_y_continuous(labels = dollar)
```



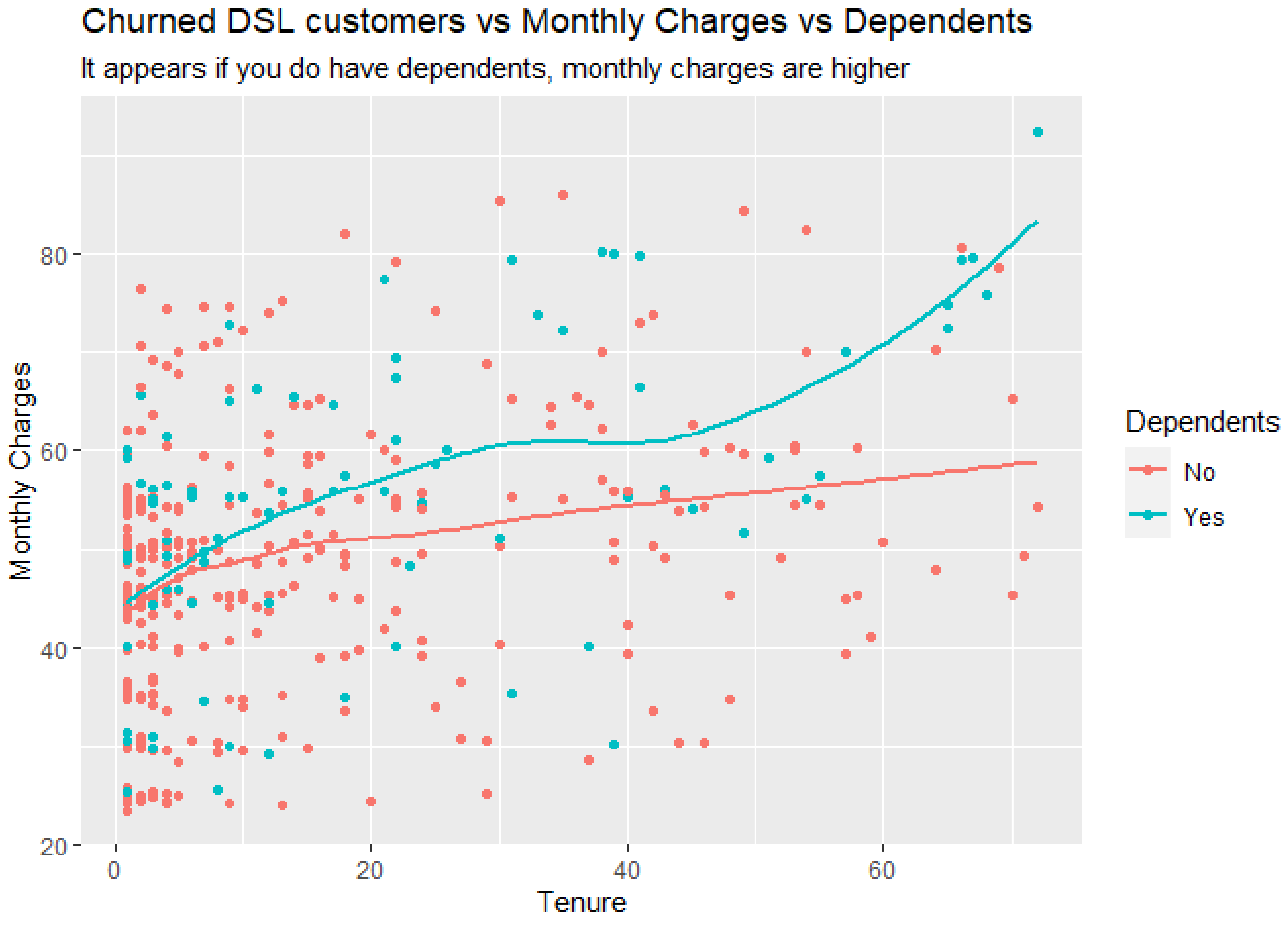
```
# How do the costs stack up?
data %>% ggplot(aes(tenure, TotalCharges, colour=InternetService))+
  geom_point()+
  ggtitle("Total Charges by Plan & Tenure")+
  labs(x="Tenure", y="Total Charges",
  subtitle = "Note the absence of longer term DSL customers")+
  scale_y_continuous(labels = dollar)
```



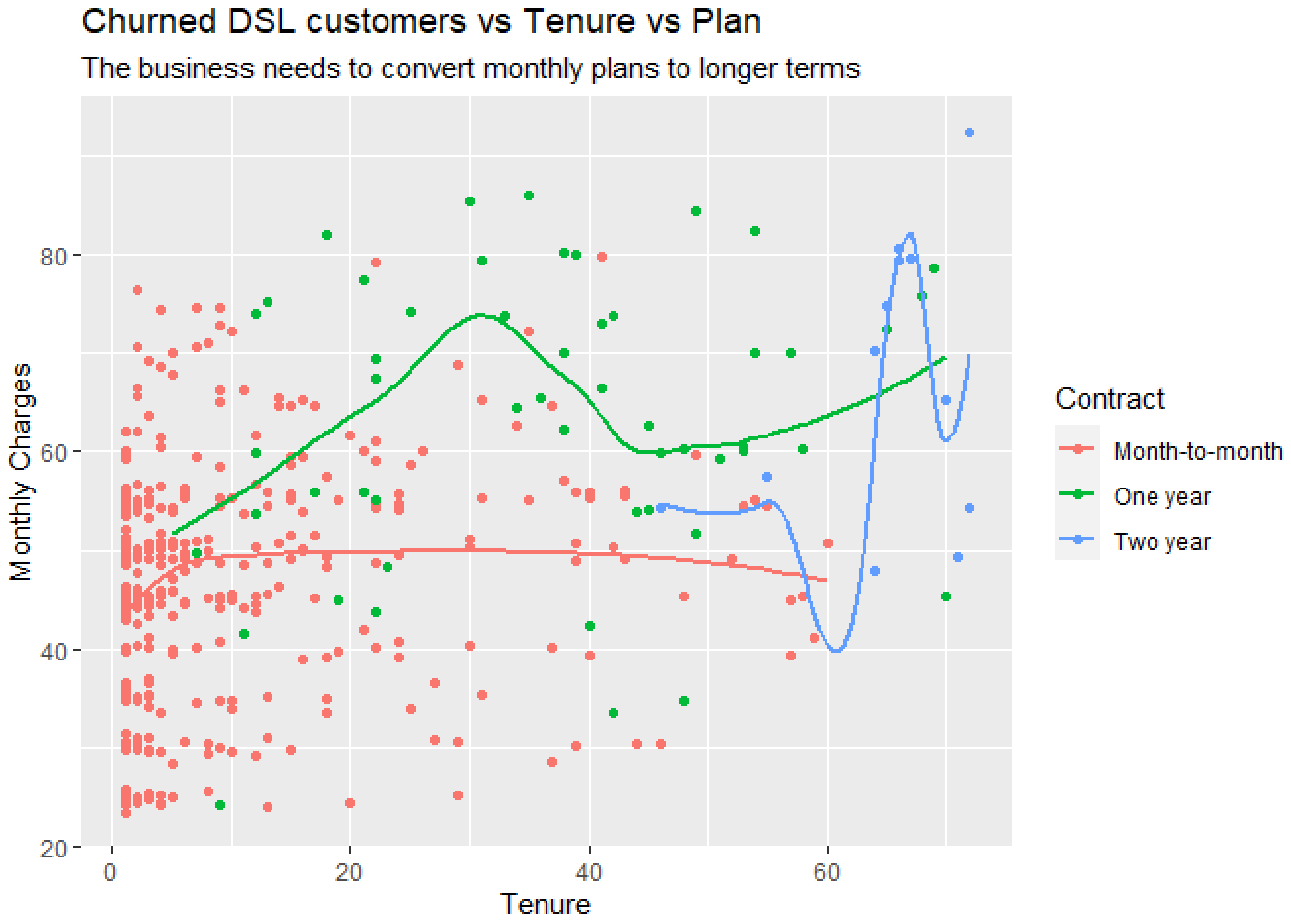
```
# Lets look more closely at this
data %>% filter(InternetService=="DSL") %>%
  ggplot(aes(tenure, MonthlyCharges, colour=Churn))+
  geom_point()+
  geom_smooth(se=FALSE)+
  ggtitle("Tenure vs Monthly Charges vs Churn")+
  labs(x="Tenure", y="Monthly Charges",
  subtitle = "DSL churn occurs early in tenure without charges correlation
```



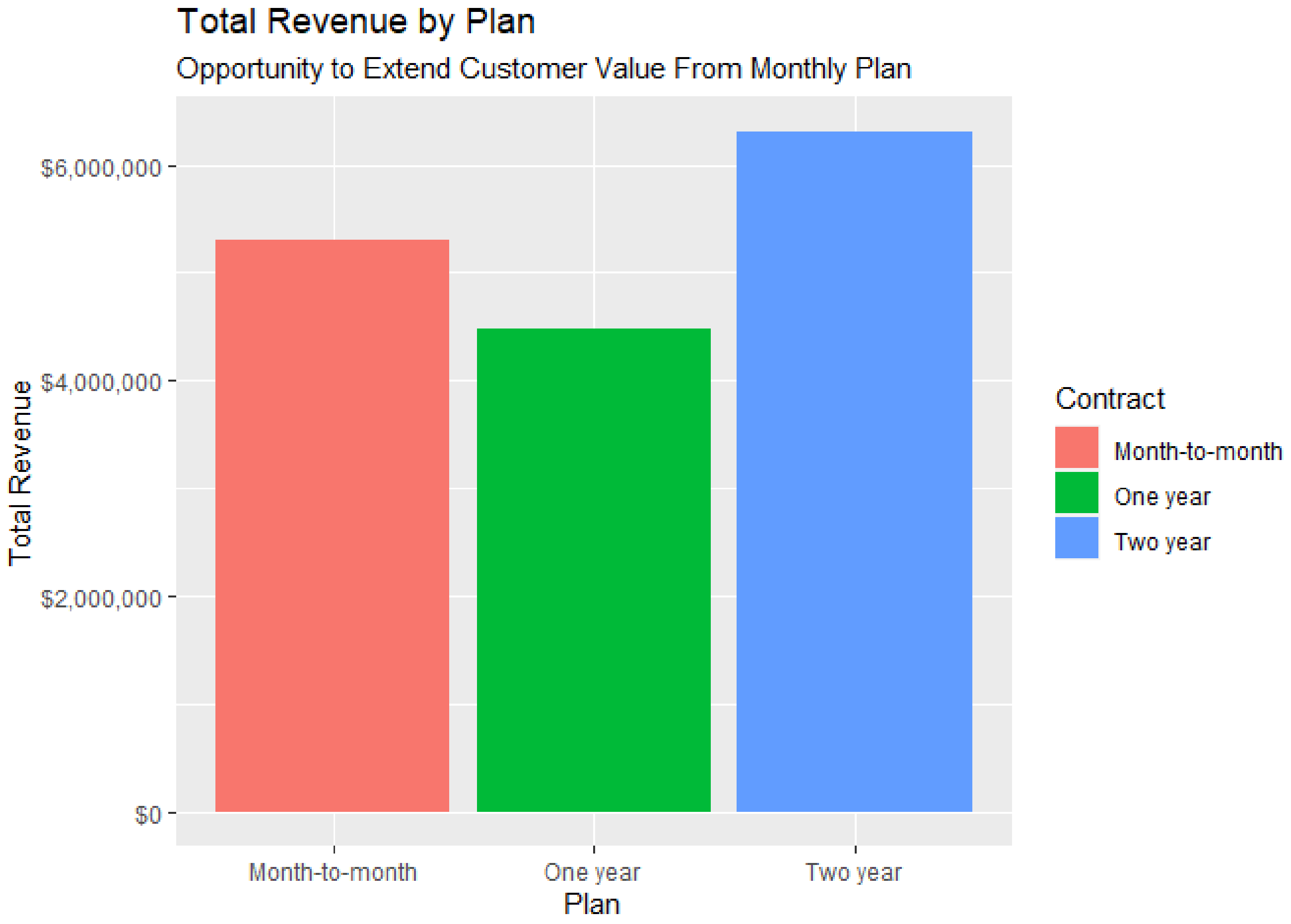
```
# Is there a connection with dependents?
data %>% filter(InternetService=="DSL" & Churn == "Yes") %>%
  ggplot(aes(tenure, MonthlyCharges, colour=Dependents))+
  geom_point()+
  geom_smooth(se=FALSE)+
  ggtitle("Churned DSL customers vs Monthly Charges vs Dependents")+
  labs(x="Tenure", y="Monthly Charges",
  subtitle="It appears if you do have dependents, monthly charges are high
```



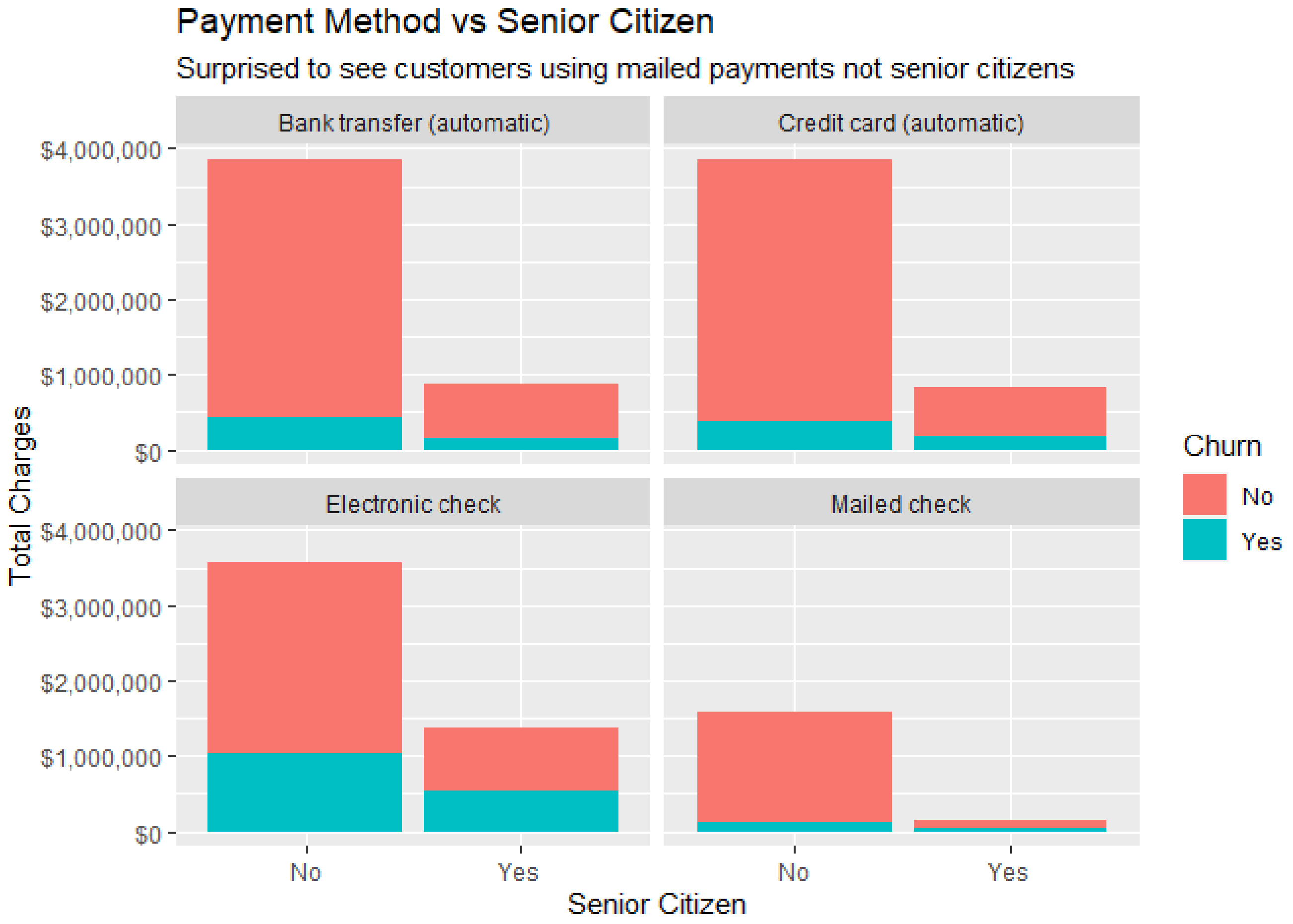
```
# Perhaps this churn is related to contract?
data %>% filter(InternetService=="DSL" & Churn == "Yes") %>%
  ggplot(aes(tenure, MonthlyCharges, colour=Contract))+
  geom_point()+
  geom_smooth(se=FALSE)+
  ggtitle("Churned DSL customers vs Tenure vs Plan")+
  labs(x="Tenure", y="Monthly Charges",
  subtitle="The business needs to convert monthly plans to longer terms")
```



```
# Customer value
data %>% group_by(Contract, TotalCharges) %>%
  ggplot(aes(Contract, TotalCharges, fill = Contract))+
  geom_col()+
  ggtitle("Total Revenue by Plan")+
  labs(x="Plan", y="Total Revenue",
       subtitle = "Opportunity to Extend Customer Value From Monthly Plan"
  scale_y_continuous(labels = dollar)
```

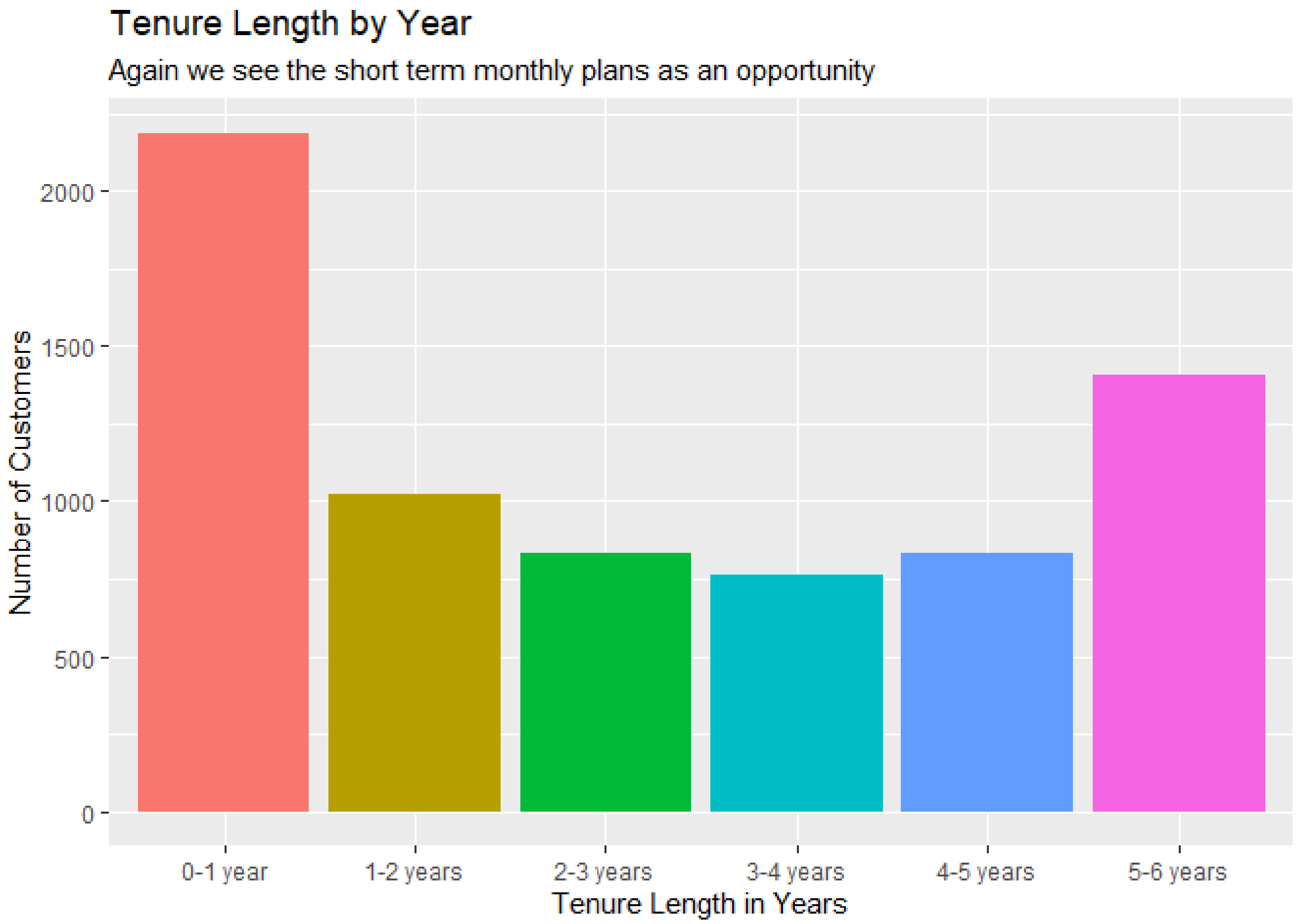


```
# Facet wrap plot
data %>% ggplot(aes(SeniorCitizen,TotalCharges, fill=Churn))+
  facet_wrap(~PaymentMethod)+
  geom_col()+
  ggtitle("Payment Method vs Senior Citizen")+
  labs(x="Senior Citizen", y="Total Charges",
       subtitle = "Surprised to see customers using mailed payments not se
  scale_y_continuous(labels = dollar)
```

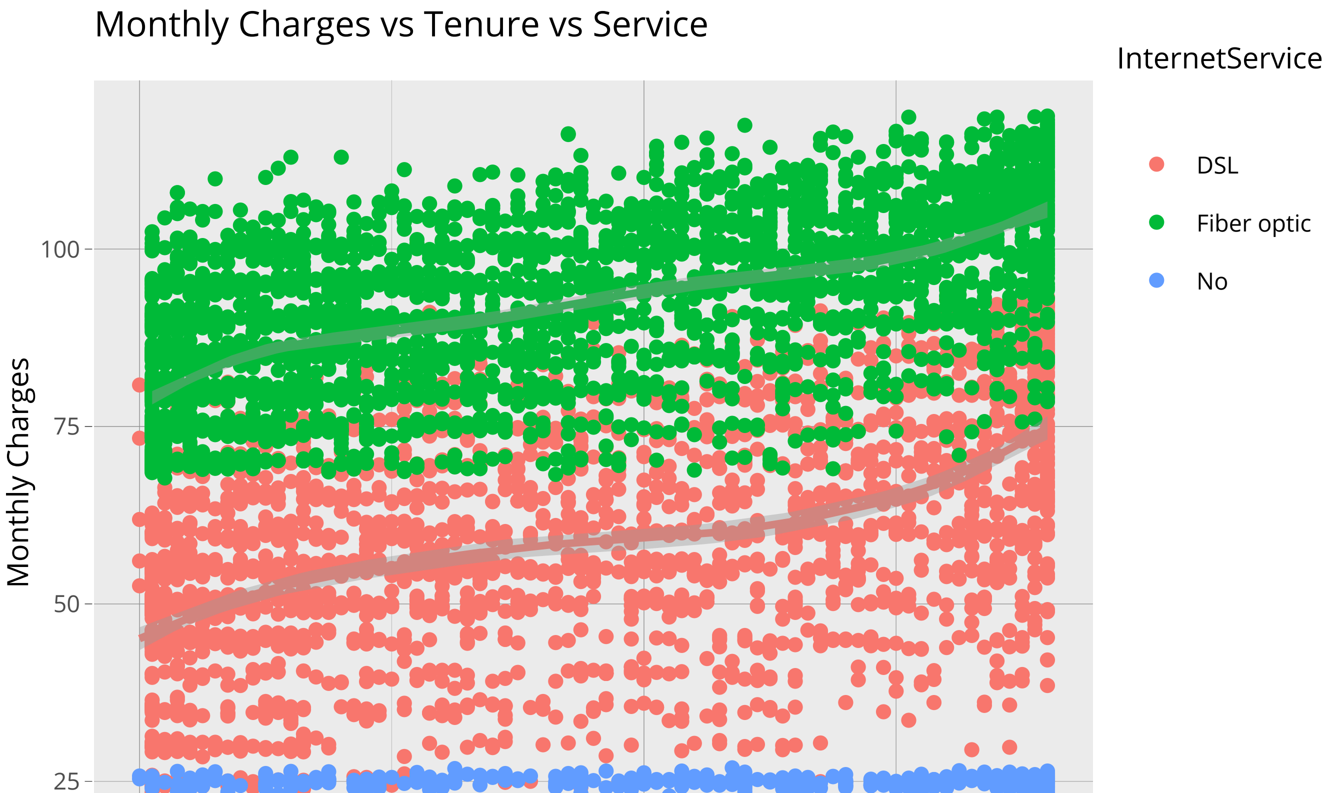


```
# Tenure length wrangling
data <- mutate(data, tenure_bin = tenure)
data$tenure_bin[data$tenure_bin >=0 & data$tenure_bin <= 12] <- '0-1 year'
data$tenure_bin[data$tenure_bin > 12 & data$tenure_bin <= 24] <- '1-2 year
data$tenure_bin[data$tenure_bin > 24 & data$tenure_bin <= 36] <- '2-3 year
data$tenure_bin[data$tenure_bin > 36 & data$tenure_bin <= 48] <- '3-4 year
data$tenure_bin[data$tenure_bin > 48 & data$tenure_bin <= 60] <- '4-5 year
data$tenure_bin[data$tenure_bin > 60 & data$tenure_bin <= 72] <- '5-6 year
data$tenure_bin <- as.factor(data$tenure_bin)

# Plot tenure length
data %>% ggplot(aes(tenure_bin, fill = tenure_bin)) +
  geom_bar()+
  ggtitle("Tenure Length by Year")+
  labs(x="Tenure Length in Years",
       y="Number of Customers",
       subtitle = "Again we see the short term monthly plans as an opportu
  scale_fill_discrete(guide=FALSE)
```



```
# Plotly object
plota <- data %>% ggplot(aes(tenure, MonthlyCharges, colour = InternetServ
  geom_point()+
  geom_smooth()+
  ggtitle("Monthly Charges vs Tenure vs Service")+
  labs(x="Tenure", y="Monthly Charges")
ggplotly(plota)
```



Feature Importance

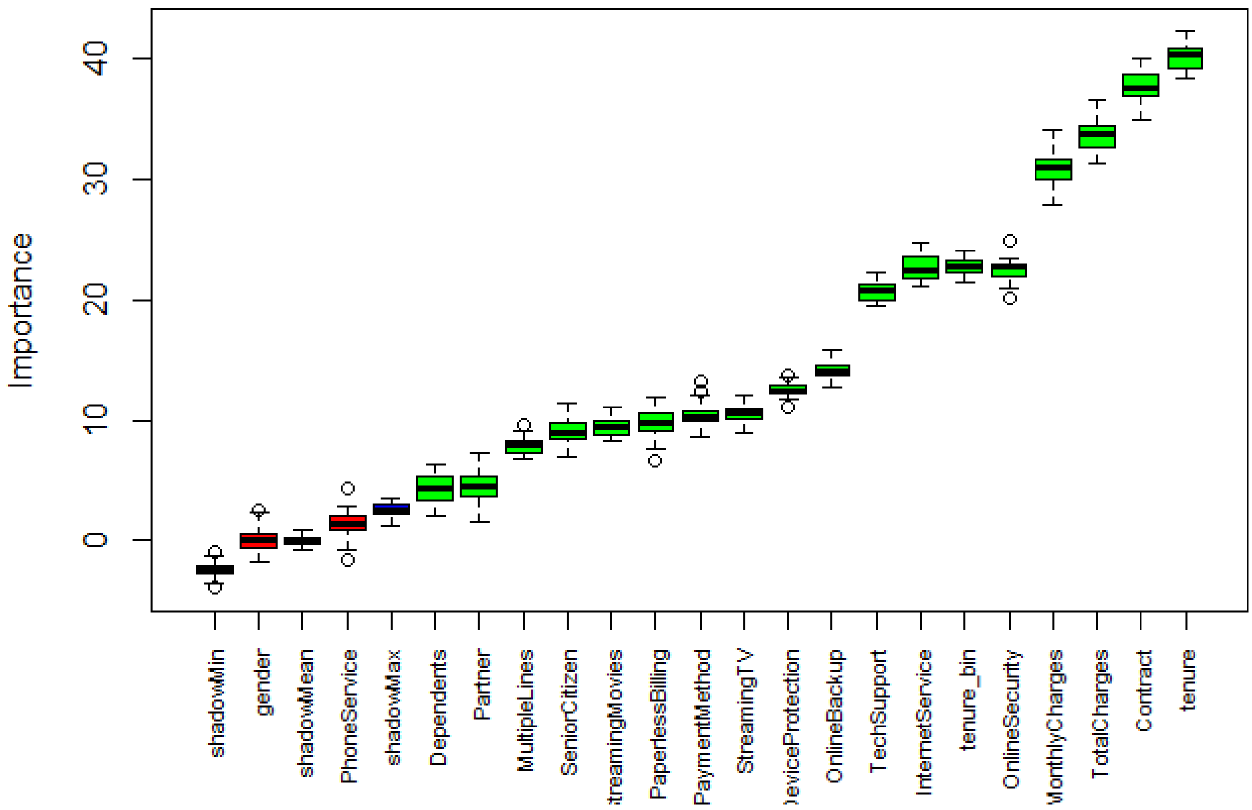
Now that we have some initial eda lets use machine learning to understand what features are the most predictive for churn and which are not. Boruta uses a powerful randomforest algorithm to calculate importance.

```r
set.seed(1)
# For ease lets remove the customer identification
boruta_data <- data[complete.cases(data[]),] %>%
  select(-customerID)

# Create and print Boruta output
boruta_output <- Boruta(Churn~., data = boruta_data)
print(boruta_output)
```

```
## Boruta performed 31 iterations in 4.334043 mins.
##  18 attributes confirmed important: Contract, Dependents,
## DeviceProtection, InternetService, MonthlyCharges and 13 more;
##  2 attributes confirmed unimportant: gender, PhoneService;
```

```r
# Tidy and plot the output
plot(boruta_output, xlab = "", xaxt = "n")
lz<-lapply(1:ncol(boruta_output$ImpHistory),function(i)
boruta_output$ImpHistory[is.finite(boruta_output$ImpHistory[,i]),i])
names(lz) <- colnames(boruta_output$ImpHistory)
Labels <- sort(sapply(lz,median))
axis(side = 1,las=2,labels = names(Labels),
at = 1:ncol(boruta_output$ImpHistory), cex.axis = 0.7)
```



```
# Get the important attributes withough tentative
getSelectedAttributes(boruta_output, withTentative = F)
```

```
##  [1] "SeniorCitizen"    "Partner"          "Dependents"       "tenure"
##  [5] "MultipleLines"    "InternetService"  "OnlineSecurity"   "OnlineBa
##  [9] "DeviceProtection" "TechSupport"      "StreamingTV"      "Streamin
## [13] "Contract"         "PaperlessBilling" "PaymentMethod"    "MonthlyC
## [17] "TotalCharges"     "tenure_bin"
```

Modelling

Now we will create training and test sets, with a caret powered randomforest and an automated machine learning package, h2o to compare results. From a business case perspective, an accurate predictive model could be used to fire targeted offers at customers with predicted churn.

```
# Remove unimportant features
data <- data %>% select(-gender, -customerID)

# Split the data into training and validation sets
test_index <- createDataPartition(data$Churn, p = .10, list = FALSE)
training <- data[-test_index,]
```

```
validation <- data[test_index,]

# Train a random forest model
rf_fit <- train(Churn ~.,
                data = training,
                method = "ranger")

# Test the model
rf_pred <- predict(rf_fit, newdata = validation, na.action = na.pass)

# Table and view the result
rf_result <- confusionMatrix(table(rf_pred,validation$Churn))
rf_result
```

```
## Confusion Matrix and Statistics
##
##
## rf_pred  No Yes
##     No  495 115
##     Yes  23  72
##
##                Accuracy : 0.8043
##                  95% CI : (0.773, 0.8329)
##     No Information Rate : 0.7348
##     P-Value [Acc > NIR] : 1.007e-05
##
##                   Kappa : 0.4042
##
##  Mcnemar's Test P-Value : 9.451e-15
##
##             Sensitivity : 0.9556
##             Specificity : 0.3850
##          Pos Pred Value : 0.8115
##          Neg Pred Value : 0.7579
##              Prevalence : 0.7348
##          Detection Rate : 0.7021
##    Detection Prevalence : 0.8652
##       Balanced Accuracy : 0.6703
##
##        'Positive' Class : No
##
```

```
# Store the accuracy for comparison later
rf.accuracy <- rf_result$overall['Accuracy']
```

Automated Machine Learning (AML)

Now we'll use an automated alogorithm that searches for the best fit model including stacked ensembles and compare results.

```
set.seed(1234)
# Use the h2o package to create a best fit stacked ensemble
h2o.init()
```

```
##  Connection successful!
## 
## R is connected to the H2O cluster: 
##     H2O cluster uptime:         22 hours 30 minutes 
##     H2O cluster timezone:       Australia/Brisbane 
##     H2O data parsing timezone:  UTC 
##     H2O cluster version:        3.30.0.1 
##     H2O cluster version age:    4 months and 25 days !!! 
##     H2O cluster name:           H2O_started_from_R_bmr057_xik111 
##     H2O cluster total nodes:    1 
##     H2O cluster total memory:   1.13 GB 
##     H2O cluster total cores:    4 
##     H2O cluster allowed cores:  4 
##     H2O cluster healthy:        TRUE 
##     H2O Connection ip:          localhost 
##     H2O Connection port:        54321 
##     H2O Connection proxy:       NA 
##     H2O Internal Security:      FALSE 
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, Targ
##     R Version:                  R version 3.6.2 (2019-12-12)
```

```
## Warning in h2o.clusterInfo(): 
## Your H2O cluster version is too old (4 months and 25 days)!
## Please download and install the latest version from http://h2o.ai/downl
```

```
# Convert data to h2o arrays
h2o_training <- as.h2o(training)
```

```
## 
  |                                                                            
  |                                                                      |
  |                                                                            
  |======================================================================|
h2o_test <- as.h2o(validation)
```

```
##
  |
  |                                                                      |
  |
  |======================================================================|
```

```
# Use the power of automated machine learning
aml <- h2o.automl(y="Churn", training_frame = h2o_training,
                  max_runtime_secs = 300)
```

```
##
  |
  |                                                                      |
## 13:45:04.639: AutoML: XGBoost is not available; skipping it.
  |
  |=====                                                                 |
  |
  |======                                                                |
  |
  |========                                                              |
  |
  |=========                                                             |
  |
  |===============                                                       |
  |
  |================                                                      |
  |
  |=====================                                                 |
  |
  |==========================                                            |
  |
  |===========================                                           |
  |
  |============================                                          |
  |
  |=============================                                         |
  |
  |==============================                                        |
  |
  |==================================================                    |
  |
  |====================================================                  |
  |
  |========================================================              |
  |
  |=========================================================             |
  |
  |==========================================================            |
  |
  |============================================================          |
  |
  |====================================================================  |
  |
  |====================================================================  |
  |
  |===================================================================== |
  |
  |===================================================================== |
  |
  |======================================================================|
```

```
aml_pred <- h2o.predict(aml@leader, h2o_test)
```

```
##
  |
  |                                                                      |
  |
  |======================================================================|
```

```
# Store accuracy and create confusion matrix
perf <- h2o.performance(aml@leader,h2o_test)
perf_cf <- h2o.confusionMatrix(perf)
h2o_acc <- max(h2o.accuracy(perf))
perf_cf
```

```
## Confusion Matrix (vertical: actual; across: predicted)  for max f1 @ th
##          No Yes    Error       Rate
## No      426  92 0.177606   =92/518
## Yes      45 142 0.240642   =45/187
## Totals 471 234 0.194326  =137/705
```

```r
# Compare best performing automated algorithm vs randomforest
overall_results <- data.frame(Method="randomforest",
                              Accuracy = rf.accuracy)
h2o_results <- data.frame(Method="aml",
                              Accuracy = h2o_acc)
overall_results <- overall_results %>%
  rbind(h2o_results) %>%  knitr::kable(row.names = FALSE)
overall_results
```

Method	Accuracy
randomforest	0.8042553
aml	0.8941511

Conclusion

It is possible to predict churn for this business with a reasonable level of accuracy and the recommendation would be to build a business case to understand the benefit of targeted promotions for churn predicted customers vs the cost of activation. The promotional cost of such an activity could be reduced by targeting customers with a higher probability of churning.

Thank you for grading this assignment and to the edX team who have created this training content. By taking the data science track I have found some new powerful skills which are a real asset for myself and any prospective employer and these skills are already being utilised.

It is clear that AML processes are very powerful also, though I could not have realised this, without having learned the science and programming background required for other methods - hard to believe such powerful tools are open source and freely available.