# Telco Churn Analysis

## Import dependencies

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(scales)
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggthemes)
library(caret)
```

```
## Loading required package: lattice
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```r
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(rpart)
library(rpart.plot)
library(precrec)
```

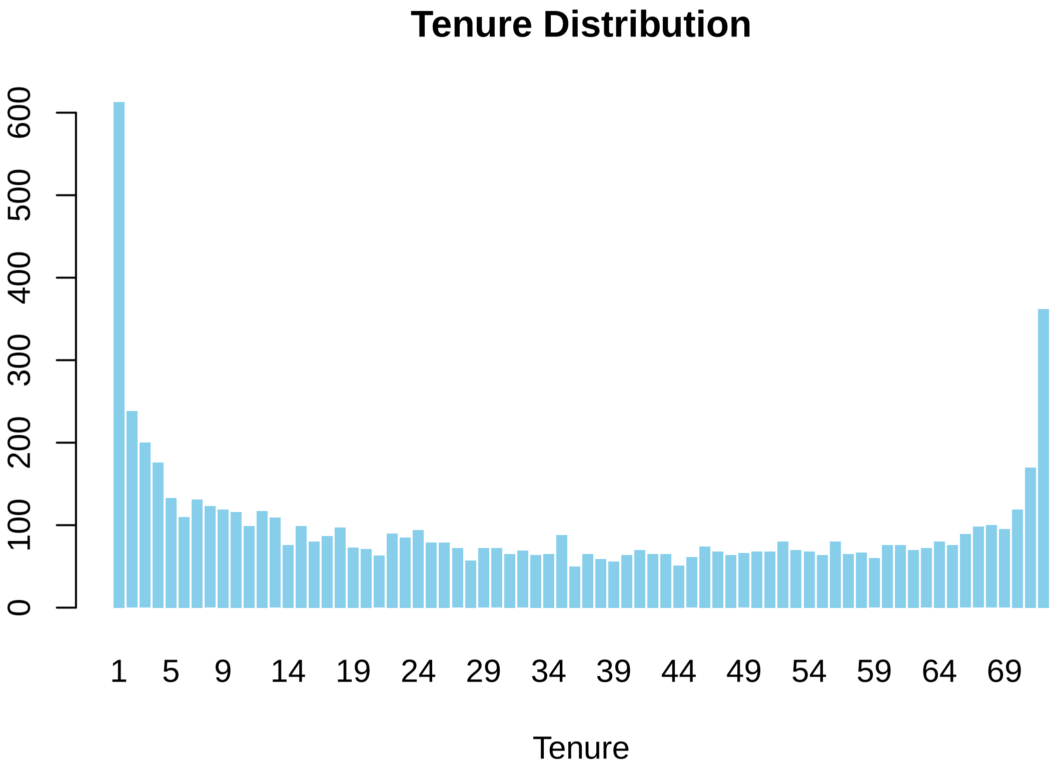
# Read in data and preprocess

```r
telcoChurn <- read.csv(file = 'telco_churn_edited.csv')
telcoChurn = subset(telcoChurn, select=-c(customerID))
telcoChurn$Churn <- as.factor(telcoChurn$Churn)
telcoChurn <- na.omit(telcoChurn)
```

# Exploratory Data Analysis

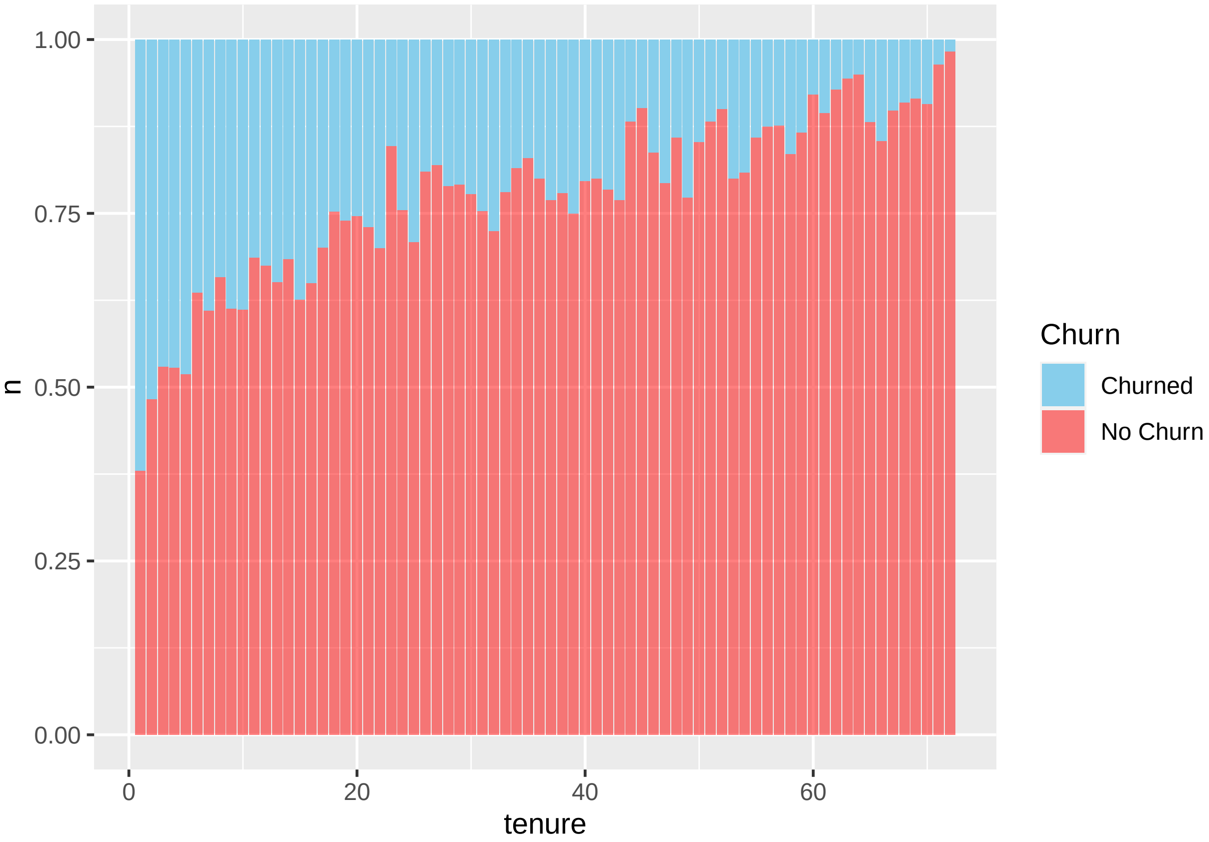
## Tenure against Churn

```
tenureCounts <- table(telcoChurn$tenure)
barplot(
  tenureCounts,
  main="Tenure Distribution",
  xlab="Tenure",
  col="skyblue",
  border=F
)
```

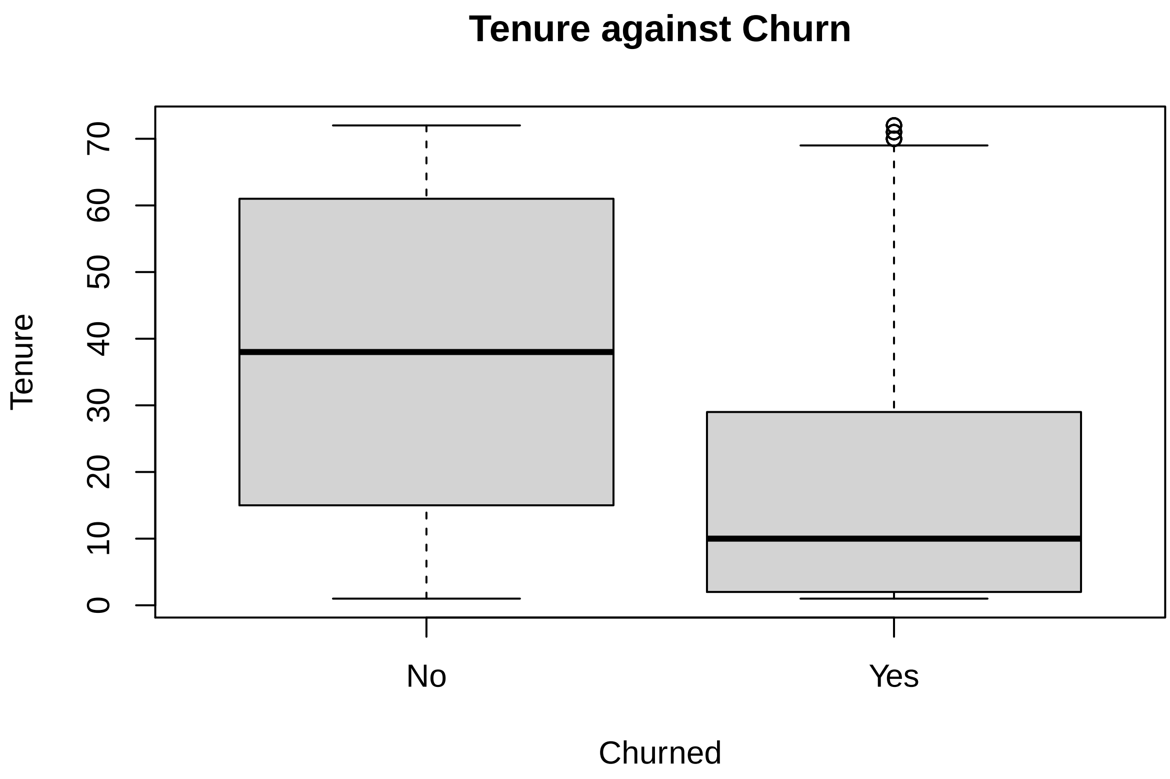


```
# percentage exited for each tenure
tenure <- telcoChurn %>% count(Churn, tenure)
tenure$Churn <- as.factor(tenure$Churn)
tenure$Churn <- ifelse(tenure$Churn == "No", "No Churn", "Churned")

ggplot(tenure, aes(fill=Churn, y=n, x=tenure))+
  geom_bar(position="fill", stat="identity")+
  scale_fill_manual(values = c("skyblue", scales::alpha("red", .5)))
```

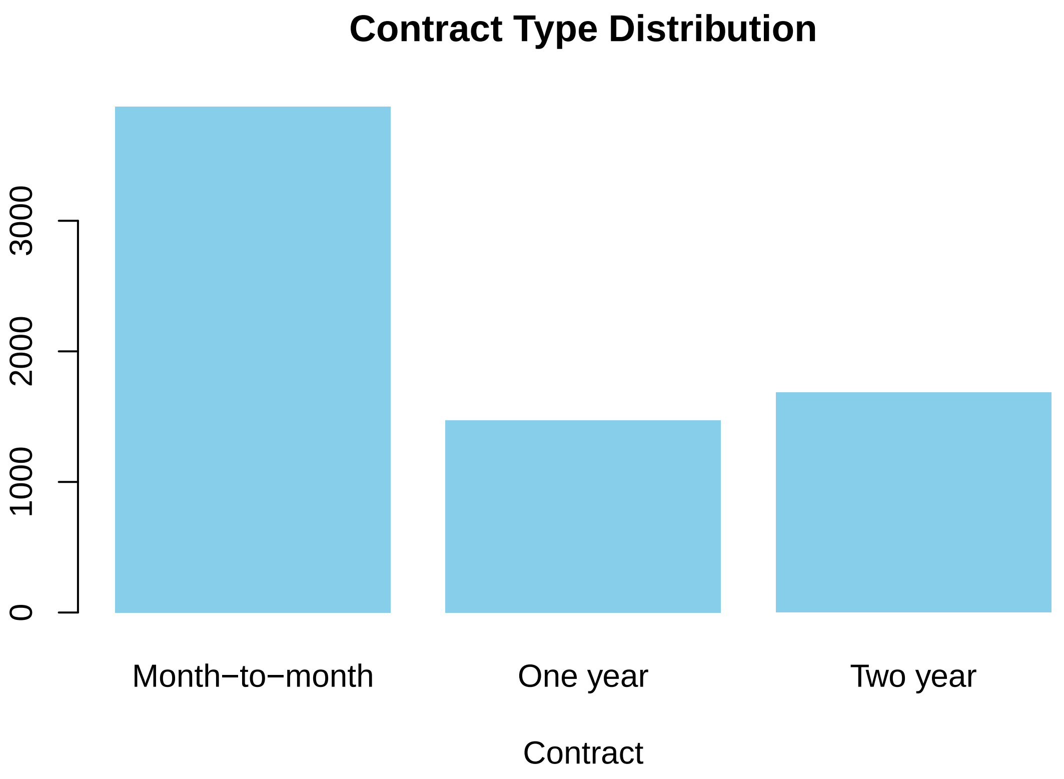


```r
# boxplot for tenure against churn
boxplot(
  tenure ~ Churn,
  data=telcoChurn,
  main="Tenure against Churn",
  xlab="Churned",
  ylab="Tenure"
  )
```



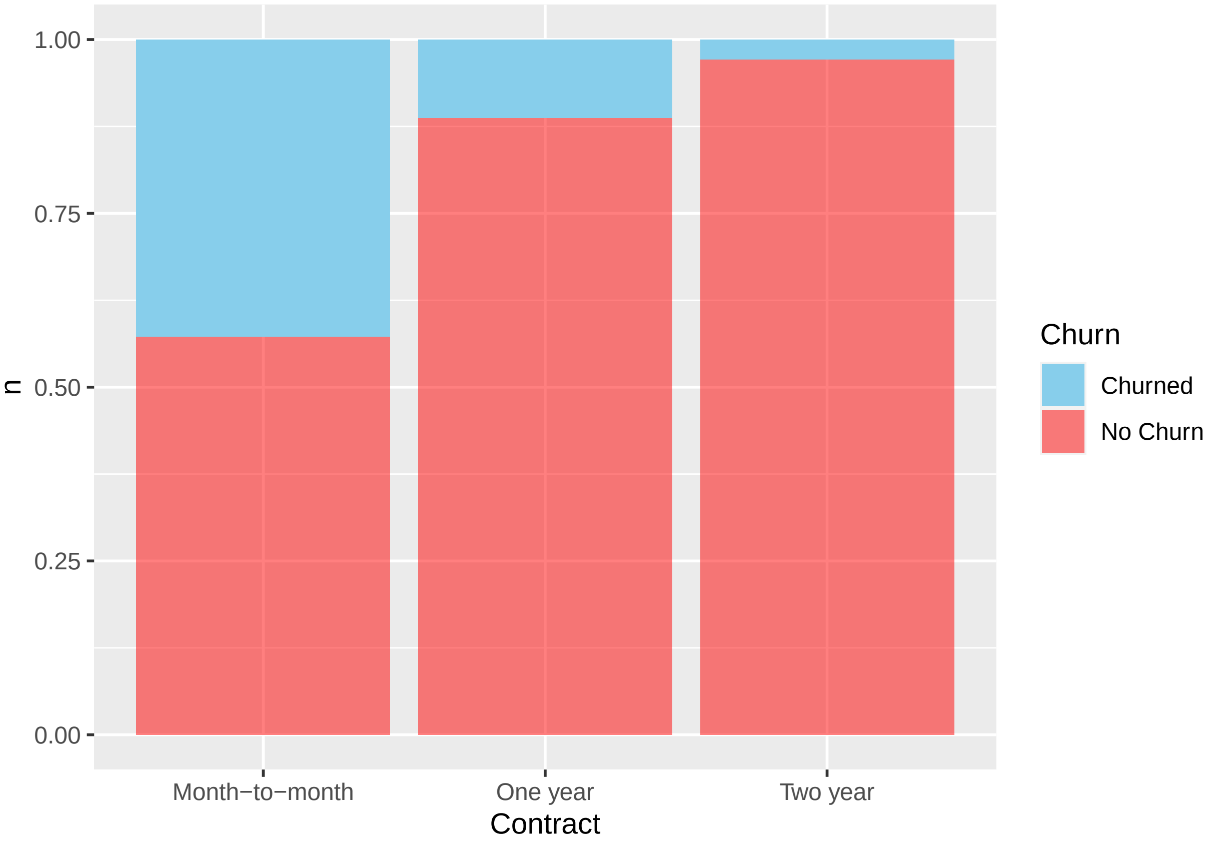
### Contract Type against Churn

```
ContractType <- table(telcoChurn$Contract)
barplot(
  ContractType,
  main="Contract Type Distribution",
  xlab="Contract",
  col="skyblue",
  border=F
)
```



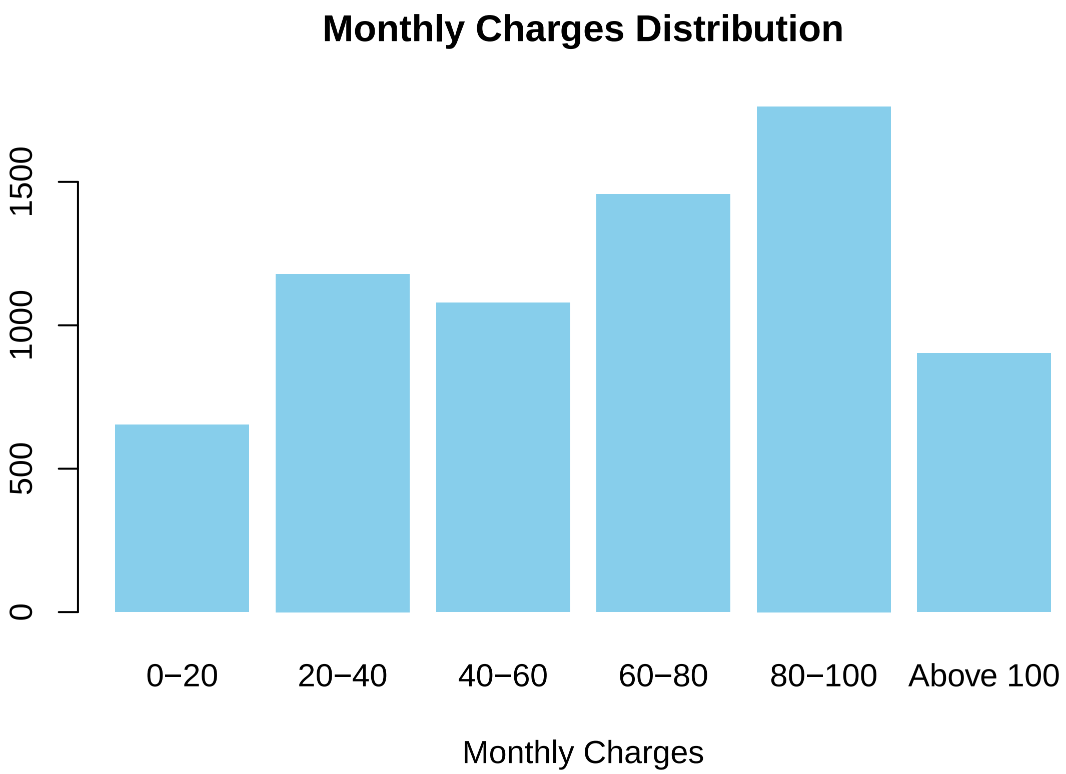
```
# percentage exited for each contract type
Contract <- telcoChurn %>% count(Churn, Contract)
Contract$Churn <- as.factor(Contract$Churn)
Contract$Churn <- ifelse(Contract$Churn == "No", "No Churn", "Churned")

ggplot(Contract, aes(fill=Churn, y=n, x=Contract))+
  geom_bar(position="fill", stat="identity")+
  scale_fill_manual(values = c("skyblue", scales::alpha("red", .5)))
```



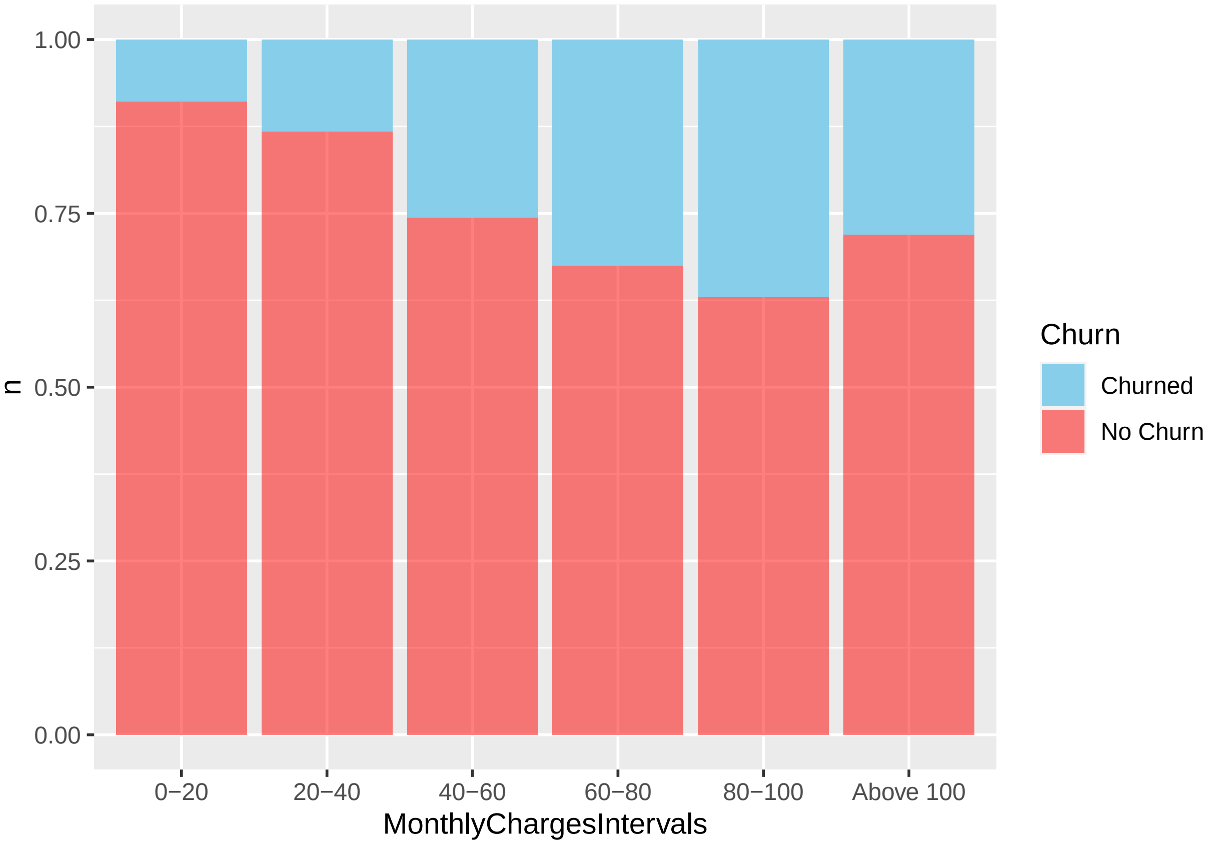
### Monthly Charges against Churn

```
MonthlyChargesCounts <- table(telcoChurn$MonthlyChargesIntervals)
barplot(
  MonthlyChargesCounts,
  main="Monthly Charges Distribution",
  xlab="Monthly Charges",
  col="skyblue",
  border=F
)
```

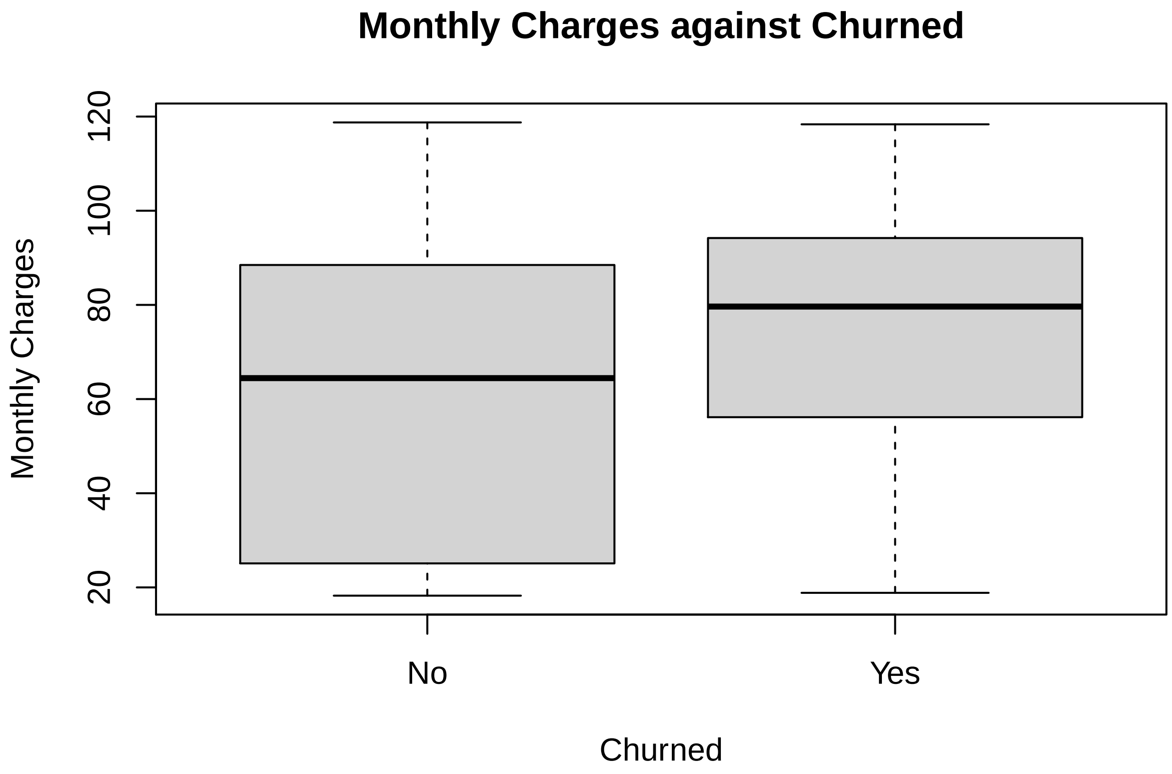


```r
# percentage exited for Montly Charge Range
MonthlyChargesIntervals <- telcoChurn %>% count(Churn, MonthlyChargesIntervals)
MonthlyChargesIntervals$Churn <- as.factor(MonthlyChargesIntervals$Churn)
MonthlyChargesIntervals$Churn <- ifelse(MonthlyChargesIntervals$Churn == "No", "No Churn", "Churned")


ggplot(MonthlyChargesIntervals, aes(fill=Churn, y=n, x=MonthlyChargesIntervals))+
  geom_bar(position="fill", stat="identity")+
  scale_fill_manual(values = c("skyblue", scales::alpha("red", .5)))
```

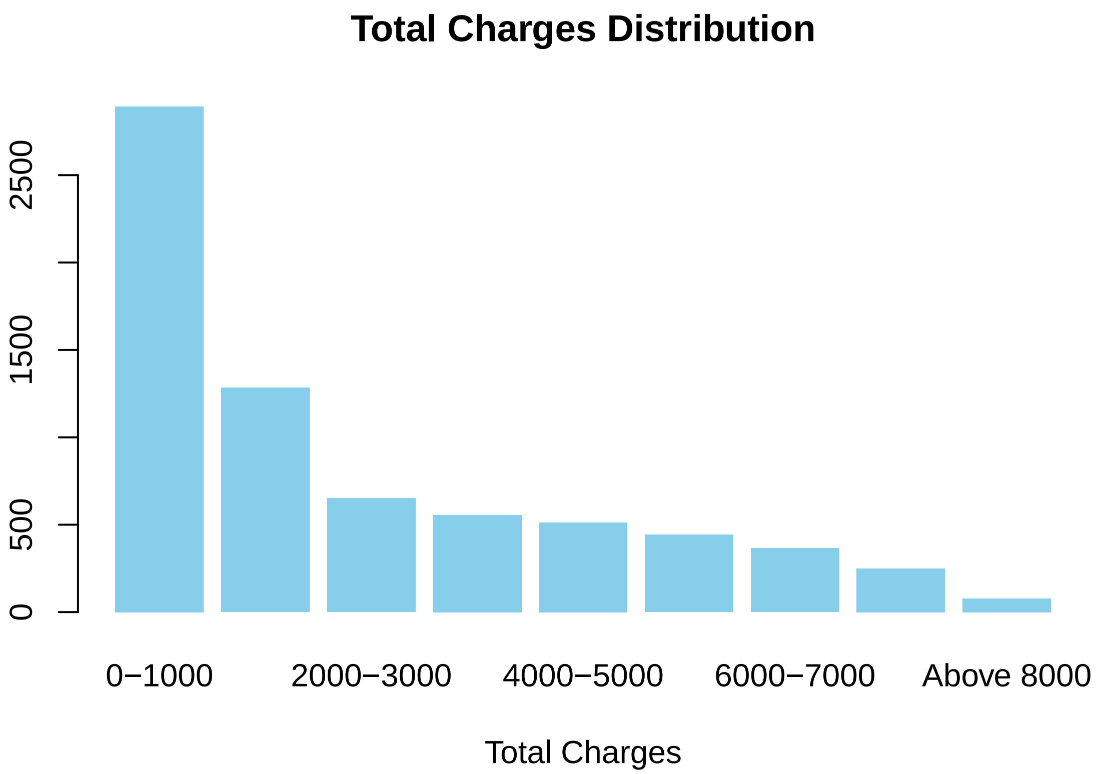


```
# boxplot for monthly charges against churn
boxplot(
  MonthlyCharges ~ Churn,
  data=telcoChurn,
  main="Monthly Charges against Churned",
  xlab="Churned",
  ylab="Monthly Charges"
  )
```



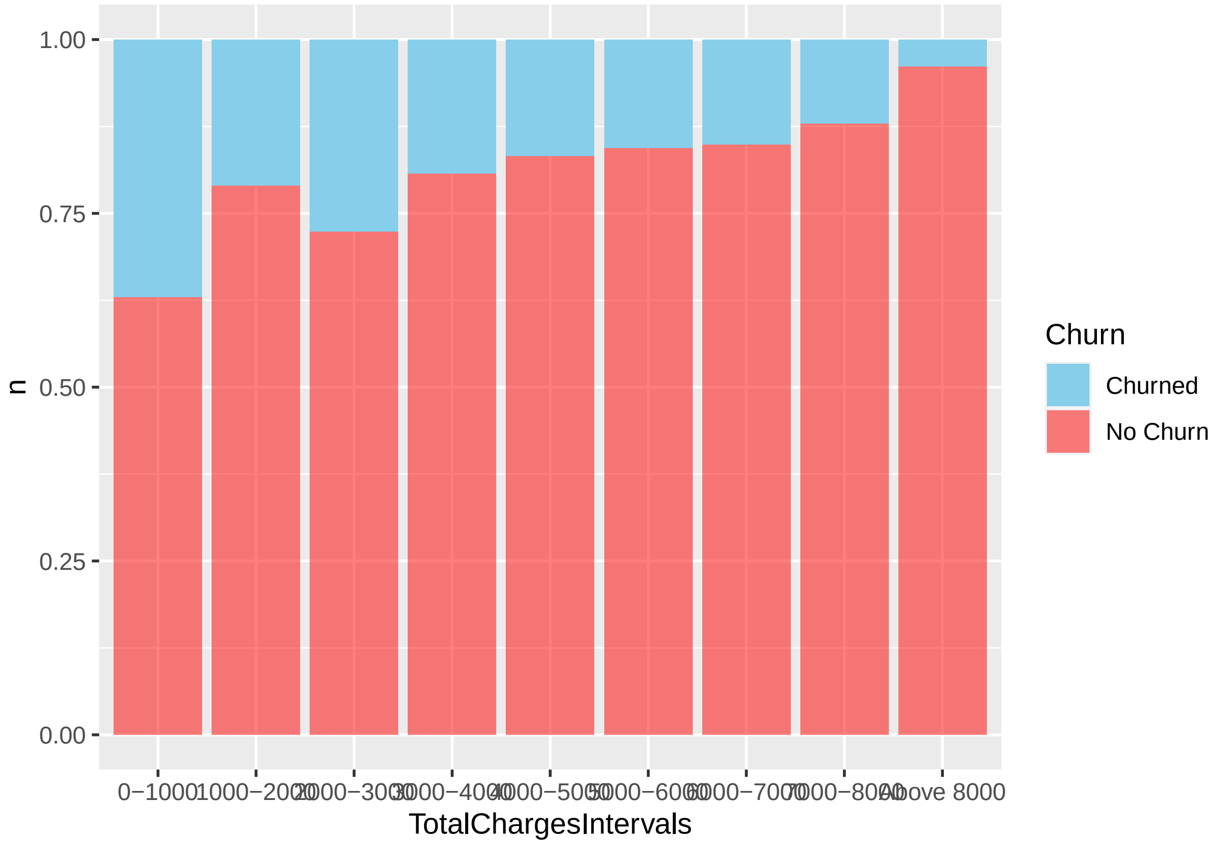
### Total Charges against Churn

```
TotalChargesCounts <- table(telcoChurn$TotalChargesIntervals)
barplot(
  TotalChargesCounts,
  main="Total Charges Distribution",
  xlab="Total Charges",
  col="skyblue",
  border=F
)
```



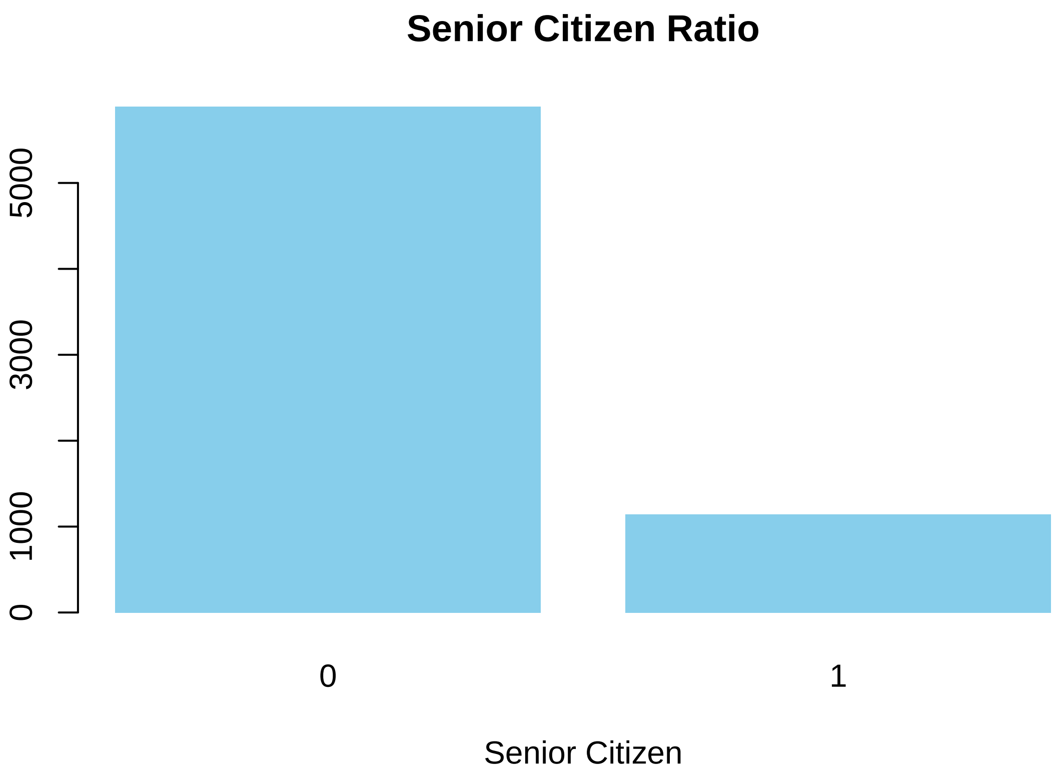
```
# percentage exited for Montly Charge Range
TotalChargesIntervals <- telcoChurn %>% count(Churn, TotalChargesIntervals)
TotalChargesIntervals$Churn <- as.factor(TotalChargesIntervals$Churn)
TotalChargesIntervals$Churn <- ifelse(TotalChargesIntervals$Churn == "No", "No Churn", "Churned")


ggplot(TotalChargesIntervals, aes(fill=Churn, y=n, x=TotalChargesIntervals))+
  geom_bar(position="fill", stat="identity")+
  scale_fill_manual(values = c("skyblue", scales::alpha("red", .5)))
```



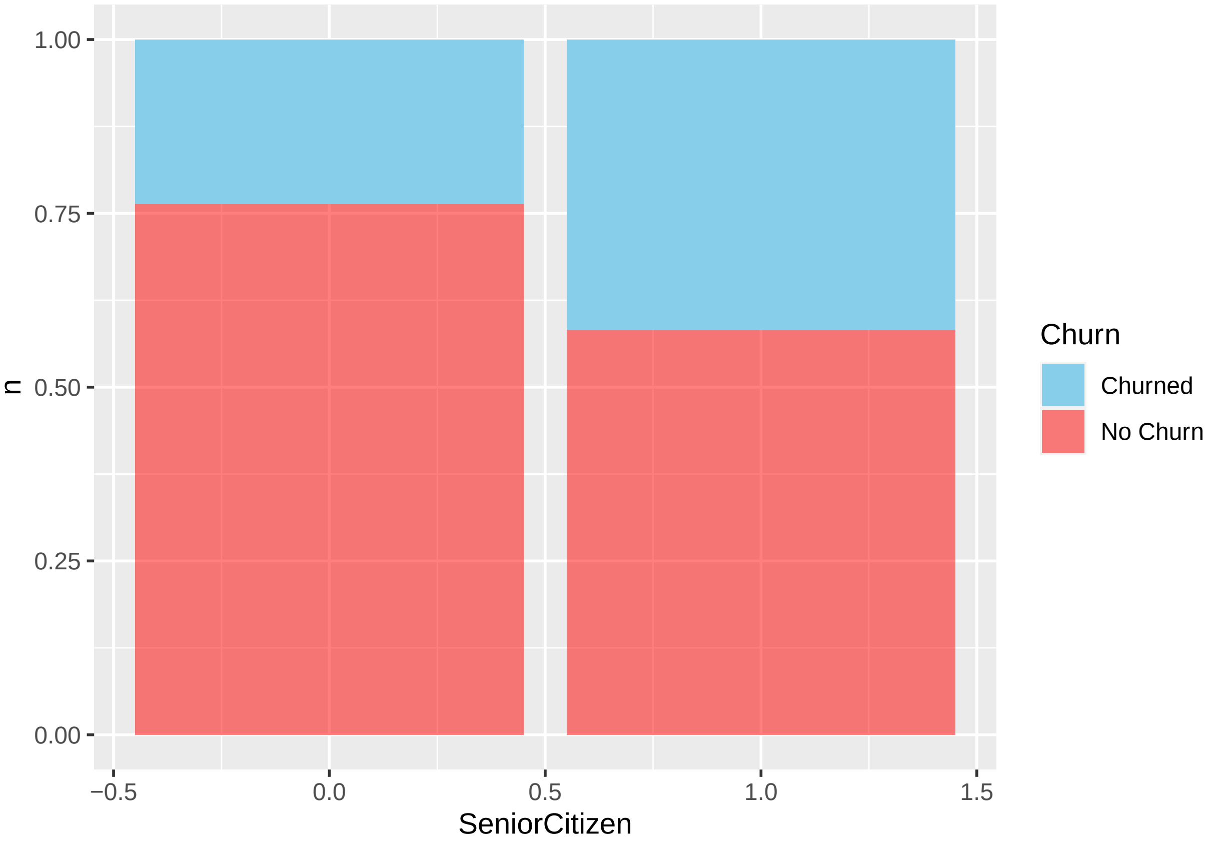
### Senior Citizen against Churn

```
SeniorCitizenscount <- table(telcoChurn$SeniorCitizen)
barplot(
  SeniorCitizenscount,
  main="Senior Citizen Ratio",
  xlab="Senior Citizen",
  col="skyblue",
  border=F
)
```



```
# percentage exited for Montly Charge Range
SeniorCitizen <- telcoChurn %>% count(Churn, SeniorCitizen)
SeniorCitizen$Churn <- as.factor(SeniorCitizen$Churn)
SeniorCitizen$Churn <- ifelse(SeniorCitizen$Churn == "No", "No Churn", "Churned")


ggplot(SeniorCitizen, aes(fill=Churn, y=n, x=SeniorCitizen))+
  geom_bar(position="fill", stat="identity")+
  scale_fill_manual(values = c("skyblue", scales::alpha("red", .5)))
```



# Training

## Train-test-split

```
# train test split
idx = createDataPartition(telcoChurn$Churn, p=0.7, list=FALSE)
set.seed(42)
train = telcoChurn[idx,]
test = telcoChurn[-idx,]

train$Churn = ifelse(train$Churn == "Yes",1,0)
test$Churn = ifelse(test$Churn == "Yes",1,0)
train$Churn = as.factor(train$Churn)
test$Churn = as.factor(test$Churn)

dim(train); dim(test)
```

```
## [1] 4924   22
```

```
## [1] 2108   22
```

## Logistic Regression

```
# logistic regression to predict churn
logreg = glm(Churn ~ .,
             family=binomial(link="logit"),
```

```r
             data=train)

print(summary(logreg))
```

```
##
## Call:
## glm(formula = Churn ~ ., family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9698  -0.6763  -0.3052   0.6772   3.1916
##
## Coefficients: (7 not defined because of singularities)
##                                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           1.0749035  1.0268513   1.047 0.295194
## genderMale                           -0.0463684  0.0774897  -0.598 0.549586
## SeniorCitizen                         0.1354064  0.1018150   1.330 0.183542
## PartnerYes                            0.0422181  0.0923358   0.457 0.647510
## DependentsYes                        -0.0805063  0.1070295  -0.752 0.451938
## tenure                               -0.0363180  0.0092040  -3.946 7.95e-05 ***
## PhoneServiceYes                       0.0265474  0.7995017   0.033 0.973511
## MultipleLinesNo phone service                NA         NA      NA       NA
## MultipleLinesYes                      0.5757736  0.2129948   2.703 0.006867 **
## InternetServiceFiber optic            1.9189455  0.9632208   1.992 0.046347 *
## InternetServiceNo                    -1.6258190  1.0090429  -1.611 0.107126
## OnlineSecurityNo internet service            NA         NA      NA       NA
## OnlineSecurityYes                    -0.1485496  0.2146207  -0.692 0.488843
## OnlineBackupNo internet service              NA         NA      NA       NA
## OnlineBackupYes                       0.0158870  0.2101957   0.076 0.939752
## DeviceProtectionNo internet service          NA         NA      NA       NA
## DeviceProtectionYes                   0.1450105  0.2108702   0.688 0.491656
## TechSupportNo internet service               NA         NA      NA       NA
## TechSupportYes                       -0.0978890  0.2158129  -0.454 0.650129
## StreamingTVNo internet service               NA         NA      NA       NA
## StreamingTVYes                        0.6220082  0.3939108   1.579 0.114323
## StreamingMoviesNo internet service           NA         NA      NA       NA
## StreamingMoviesYes                    0.5565557  0.3912933   1.422 0.154925
## ContractOne year                     -0.5344155  0.1282571  -4.167 3.09e-05 ***
## ContractTwo year                     -1.1114079  0.2038995  -5.451 5.02e-08 ***
## PaperlessBillingYes                   0.3707521  0.0897391   4.131 3.60e-05 ***
## PaymentMethodCredit card (automatic) -0.1003492  0.1357758  -0.739 0.459858
## PaymentMethodElectronic check         0.2993887  0.1135679   2.636 0.008384 **
## PaymentMethodMailed check            -0.1238955  0.1377694  -0.899 0.368495
## MonthlyCharges                       -0.0352845  0.0397288  -0.888 0.374468
## MonthlyChargesIntervals20-40         -0.1583337  0.2487169  -0.637 0.524384
## MonthlyChargesIntervals40-60          0.2831188  0.3976267   0.712 0.476451
## MonthlyChargesIntervals60-80         -0.0087914  0.5125293  -0.017 0.986315
## MonthlyChargesIntervals80-100        -0.0399477  0.6116405  -0.065 0.947925
## MonthlyChargesIntervalsAbove 100      0.2140849  0.7088860   0.302 0.762651
## TotalCharges                         -0.0007546  0.0002023  -3.730 0.000192 ***
## TotalChargesIntervals1000-2000        0.4057784  0.1914898   2.119 0.034085 *
## TotalChargesIntervals2000-3000        1.3437144  0.3238869   4.149 3.34e-05 ***
## TotalChargesIntervals3000-4000        2.2043882  0.4640929   4.750 2.04e-06 ***
## TotalChargesIntervals4000-5000        2.9534158  0.6147289   4.804 1.55e-06 ***
## TotalChargesIntervals5000-6000        3.9029024  0.7637289   5.110 3.22e-07 ***
## TotalChargesIntervals6000-7000        4.8475966  0.9177026   5.282 1.28e-07 ***
## TotalChargesIntervals7000-8000        5.5574340  1.0701044   5.193 2.07e-07 ***
## TotalChargesIntervalsAbove 8000       5.6977092  1.3302252   4.283 1.84e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5702.8  on 4923  degrees of freedom
## Residual deviance: 4110.9  on 4887  degrees of freedom
## AIC: 4184.9
##
## Number of Fisher Scoring iterations: 6
```

### Feature importance using deviance

```
# feature importance: the steeper the drop in deviance the more important the feature
anova(logreg, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Churn
##
## Terms added sequentially (first to last)
##
##
##                         Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                     4923     5702.8
## gender                   1     0.75      4922     5702.0  0.385995
## SeniorCitizen            1    82.82      4921     5619.2 < 2.2e-16 ***
## Partner                  1   109.76      4920     5509.4 < 2.2e-16 ***
## Dependents               1    29.51      4919     5479.9 5.563e-08 ***
## tenure                   1   591.08      4918     4888.8 < 2.2e-16 ***
## PhoneService             1     0.28      4917     4888.6  0.594329
## MultipleLines            1   128.27      4916     4760.3 < 2.2e-16 ***
## InternetService          2   427.35      4914     4332.9 < 2.2e-16 ***
## OnlineSecurity           1    26.00      4913     4306.9 3.415e-07 ***
## OnlineBackup             1     1.11      4912     4305.8  0.292842
## DeviceProtection         1     0.09      4911     4305.7  0.759166
## TechSupport              1    17.68      4910     4288.1 2.613e-05 ***
## StreamingTV              1    19.62      4909     4268.4 9.449e-06 ***
## StreamingMovies          1     6.36      4908     4262.1  0.011643 *
## Contract                 2    44.95      4906     4217.1 1.732e-10 ***
## PaperlessBilling         1    18.47      4905     4198.7 1.728e-05 ***
## PaymentMethod            3    23.04      4902     4175.6 3.970e-05 ***
## MonthlyCharges           1     0.76      4901     4174.9  0.383383
## MonthlyChargesIntervals  5    18.16      4896     4156.7  0.002756 **
## TotalCharges             1     9.05      4895     4147.7  0.002628 **
## TotalChargesIntervals    8    36.77      4887     4110.9 1.267e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Accuracy

```
# evaluating logistic regression model against test data
logreg.scores <- predict(logreg,newdata=test,type='response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
logreg.pred <- ifelse(logreg.scores > 0.5,1,0)
misclassError <- mean(logreg.pred != test$Churn)
print(paste('Logistic Regression Accuracy',1-misclassError))
```

```
## [1] "Logistic Regression Accuracy 0.805028462998102"
```

### Confusion Matrix

```
print("Confusion Matrix for Logistic Regression"); table(Predicted = logreg.pred, Actual = test$Churn)
```

```
## [1] "Confusion Matrix for Logistic Regression"
```

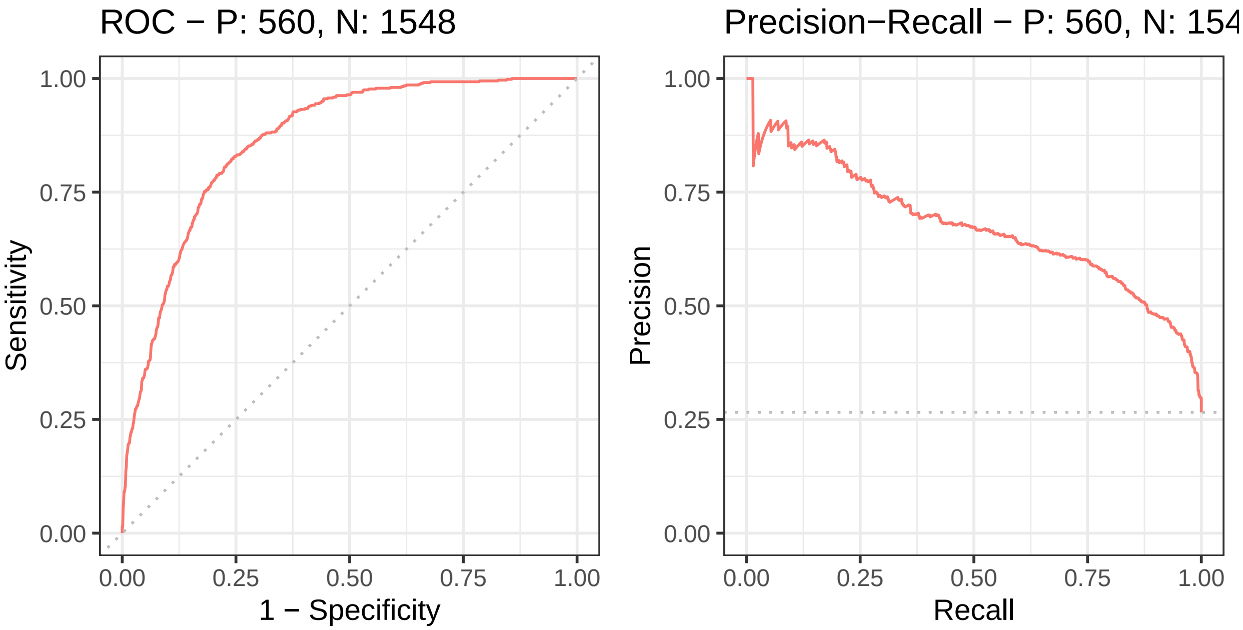
```
##          Actual
## Predicted    0    1
##         0 1398  261
##         1  150  299
```

### ROC, PRC curves

```
precrec.logreg <- evalmod(scores = logreg.scores, labels = test$Churn)
print(precrec.logreg)
```

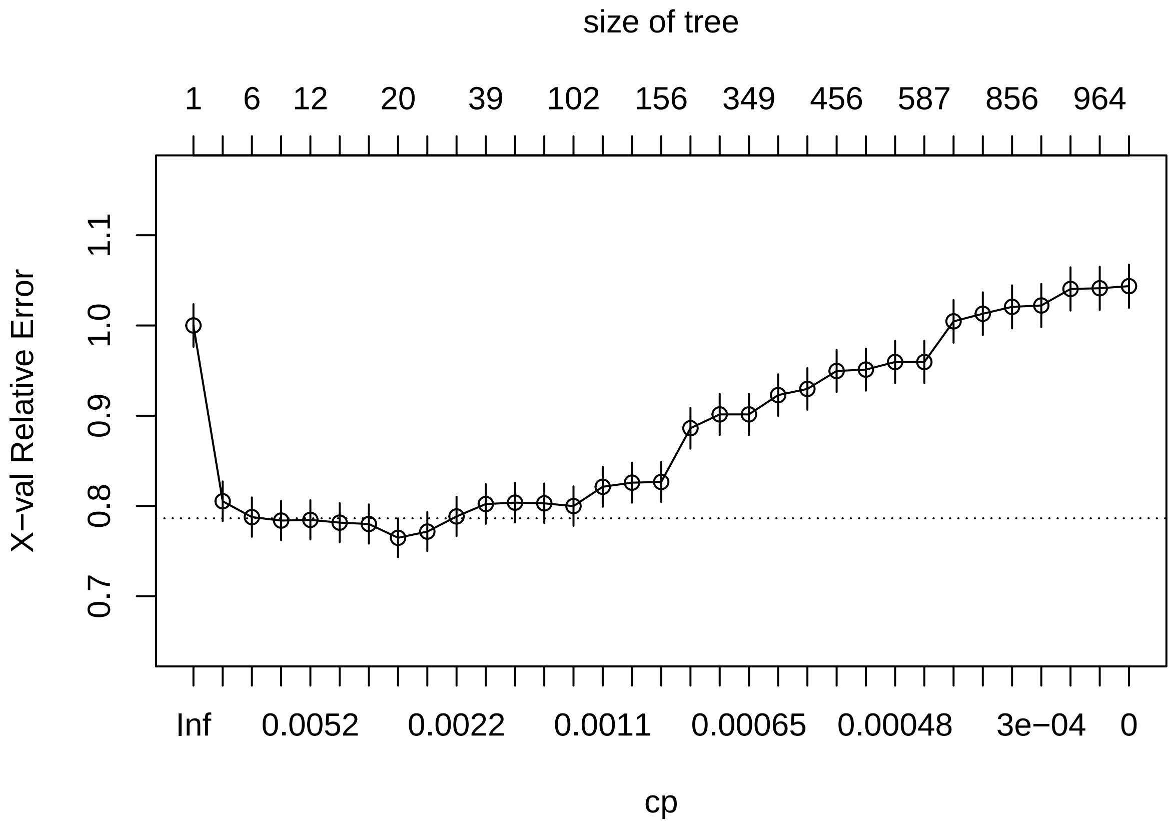
```
##
##     === AUCs ===
##
##      Model name Dataset ID Curve type       AUC
##    1         m1          1        ROC 0.8638774
##    2         m1          1        PRC 0.6755818
##
##
##     === Input data ===
##
##      Model name Dataset ID # of negatives # of positives
##    1         m1          1           1548            560
```

```
autoplot(precrec.logreg)
```



### Decision Tree

```
cart <- rpart(Churn ~ .,
              data=train,
              method = 'class',
              control=rpart.control(minsplit = 2, cp = 0)
              )
plotcp(cart)
```



### Optimisation of CP value

```
CVerror.cap <- cart$cptable[which.min(cart$cptable[,"xerror"]), "xerror"] + cart$cptable[which.min(cart

# Find the optimal CP region whose CV error is just below CVerror.cap in maximal tree cart1.
i <- 1; j<- 4
while (cart$cptable[i,j] > CVerror.cap) {
  i <- i + 1
}

# Get geometric mean of the two identified CP values in the optimal region if optimal tree has at least
cp.opt = ifelse(i > 1, sqrt(cart$cptable[i,1] * cart$cptable[i-1,1]), 1)
```

```
cart.opt <- prune(cart, cp = cp.opt)
cart.opt$variable.importance
```

```
##                Contract                  tenure   TotalChargesIntervals
##            291.35285121            271.05518868            166.01119860
##          OnlineSecurity            TotalCharges             TechSupport
##            142.37651359            130.80241298            125.73512692
##        DeviceProtection          MonthlyCharges         InternetService
##            108.65083704            105.08046232             99.09262490
## MonthlyChargesIntervals           MultipleLines             StreamingTV
##             90.43297869             46.27607262             16.63781288
##            OnlineBackup           PaymentMethod        PaperlessBilling
##             16.60431846              5.03950863              2.28343428
##              Dependents                 Partner
##              0.08954644              0.04477322
```

### Accuracy

```
tree.pred <- predict(cart.opt, test, type="class")
tree.pred.scores <- predict(cart.opt, test, type="prob")
table.pred <- table(Predicted = tree.pred, Actual = test$Churn)
print(paste('Decision Tree Accuracy',sum(diag(table.pred))/sum(table.pred)))
```

```
## [1] "Decision Tree Accuracy 0.790322580645161"
```

### Confusion Matrix

```
# Note: accuracy is 80% because of unbalanced dataset; most data points have Churn = 0. From the confus
print("Confusion Matrix for Decision Tree"); table(Predicted = tree.pred, Actual = test$Churn)
```

```
## [1] "Confusion Matrix for Decision Tree"
```

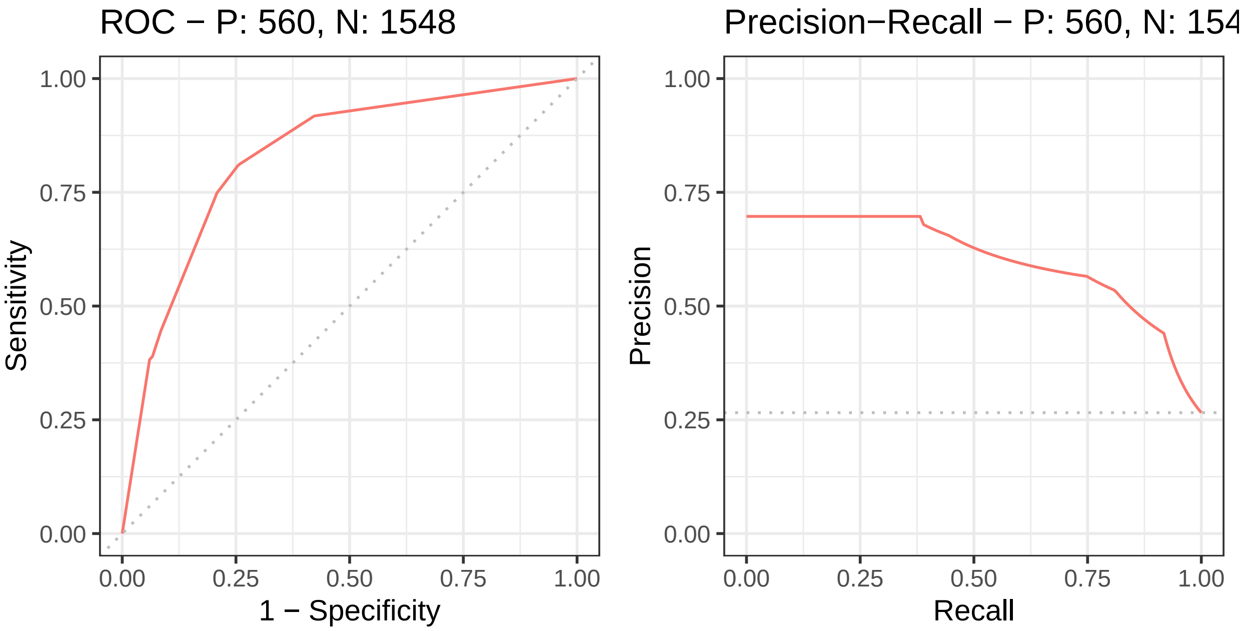
```
##          Actual
## Predicted    0    1
##         0 1417  311
##         1  131  249
```

### ROC, PRC curves

```
precrec.tree <- evalmod(scores = tree.pred.scores[, 2], labels = test$Churn)
print(precrec.tree)
```

```
##
##     === AUCs ===
##
##      Model name Dataset ID Curve type       AUC
##    1         m1          1        ROC 0.8301483
##    2         m1          1        PRC 0.6039751
##
##
##     === Input data ===
##
##      Model name Dataset ID # of negatives # of positives
##    1         m1          1           1548            560
```

```
autoplot(precrec.tree)
```



### Random Forest

```
rfmodel <- randomForest(Churn ~ ., data = train, proximity = TRUE)
```

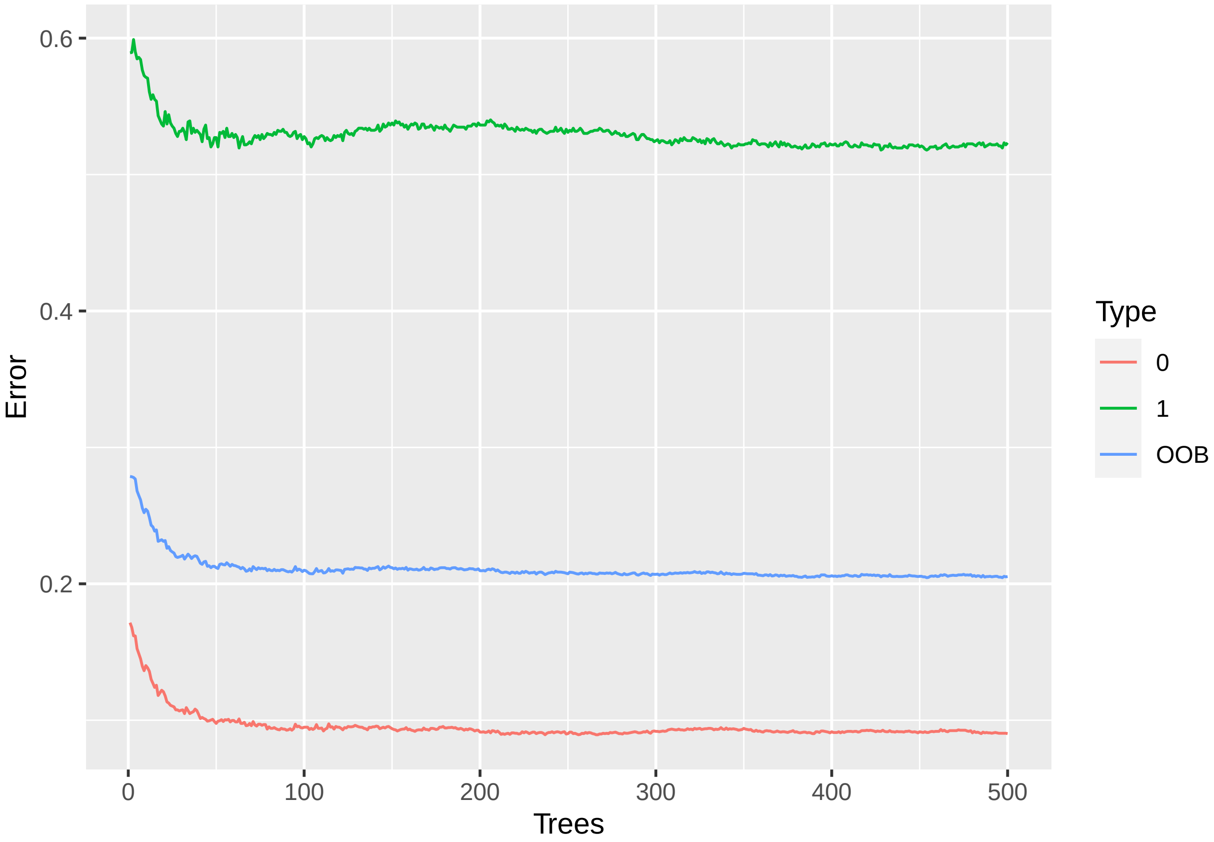
```
rfmodel
```

```
##
## Call:
##  randomForest(formula = Churn ~ ., data = train, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 20.53%
## Confusion matrix:
##      0   1 class.error
## 0 3289 326  0.09017981
## 1  685 624  0.52330023
```

#### OOB error plot for initial model

```
# model based on err.rate matrix: [OOB, No, Yes]
oob.error.data <- data.frame(Trees=rep(1:nrow(rfmodel$err.rate), times=3), Type=rep(c("OOB", "0", "1"),

ggplot(data=oob.error.data, aes(x=Trees, y=Error))+geom_line(aes(color=Type))
```



### Accuracy for initial model

```
rfmodel.pred <- predict(rfmodel, test, type="response")
rfmodel.pred.scores <- predict(rfmodel, test, type="prob")
table.rfmodel.pred <- table(Predicted = rfmodel.pred, Actual = test$Churn)
print(paste('Random Forest Accuracy',sum(diag(table.rfmodel.pred))/sum(table.rfmodel.pred)))
```

```
## [1] "Random Forest Accuracy 0.799810246679317"
```

### Confusion Matrix for initial model

```
print("Confusion Matrix for Random Forest"); table(Predicted = rfmodel.pred, Actual = test$Churn)
```

```
## [1] "Confusion Matrix for Random Forest"
```

```
##          Actual
## Predicted    0    1
##         0 1410  284
##         1  138  276
```

### ROC, PRC curves for initial model

```
precrec.rf <- evalmod(scores = rfmodel.pred.scores[, 2], labels = test$Churn)
print(precrec.rf)
```

```
##
##     === AUCs ===
##
##     Model name Dataset ID Curve type       AUC
##   1         m1          1        ROC 0.8498823
##   2         m1          1        PRC 0.6640752
##
##
##    === Input data ===
##
##     Model name Dataset ID # of negatives # of positives
##   1         m1          1           1548            560
```

### Optimisation of no. of variable in each internal node

```
# optimize no. of variables at each internal node in tree
oob.values <- vector(length=10)
for(i in 1:10){
  temp.model <- randomForest(Churn ~ ., data = telcoChurn, mtry=i, ntree=1000)

  #store OOB error rate for each random forest that uses diff value of i
  oob.values[i] <- temp.model$err.rate[nrow(temp.model$err.rate),1]
}

oob.values
```

```
##  [1] 0.2127418 0.2006542 0.2037827 0.2025028 0.2044937 0.2044937 0.2070535
##  [8] 0.2087600 0.2076223 0.2100398
```

```
# no. of variables = 2 gives lowest oob err.rate

rfmodeloptim <- randomForest(Churn ~ ., data = train, mtry=2, proximity = TRUE, type='classification')
```

```
rfmodeloptim
```

```
##
## Call:
##  randomForest(formula = Churn ~ ., data = train, mtry = 2, proximity = TRUE,      type = "classifica
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 20.09%
## Confusion matrix:
##      0   1 class.error
## 0 3315 300  0.08298755
## 1  689 620  0.52635600
```

```
# error rate reduced
```

### Accuracy for optimised model

```
rfmodeloptim.pred <- predict(rfmodeloptim, test, type="response")
rfmodeloptim.pred.scores <- predict(rfmodeloptim, test, type="prob")
table.rfmodeloptim.pred <- table(Predicted = rfmodeloptim.pred, Actual = test$Churn)
print(paste('Random Forest Accuracy',sum(diag(table.rfmodeloptim.pred))/sum(table.rfmodeloptim.pred)))
```

```
## [1] "Random Forest Accuracy 0.796489563567362"
```

### Confusion Matrix for optimised model

```
print("Confusion Matrix for Random Forest"); table(Predicted = rfmodeloptim.pred, Actual = test$Churn)
```

```
## [1] "Confusion Matrix for Random Forest"
```

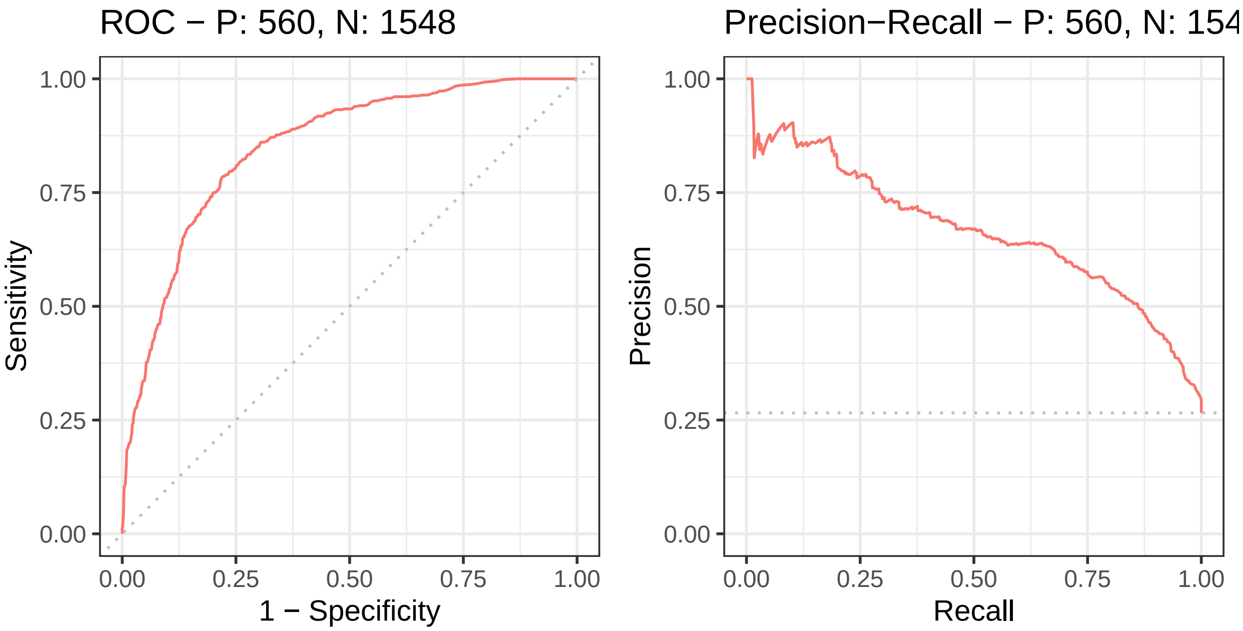
```
##          Actual
## Predicted    0    1
##         0 1421  302
##         1  127  258
```

### ROC, PRC curves for optimised model

```
precrec.rf <- evalmod(scores = rfmodeloptim.pred.scores[, 2], labels = test$Churn)
print(precrec.rf)
```

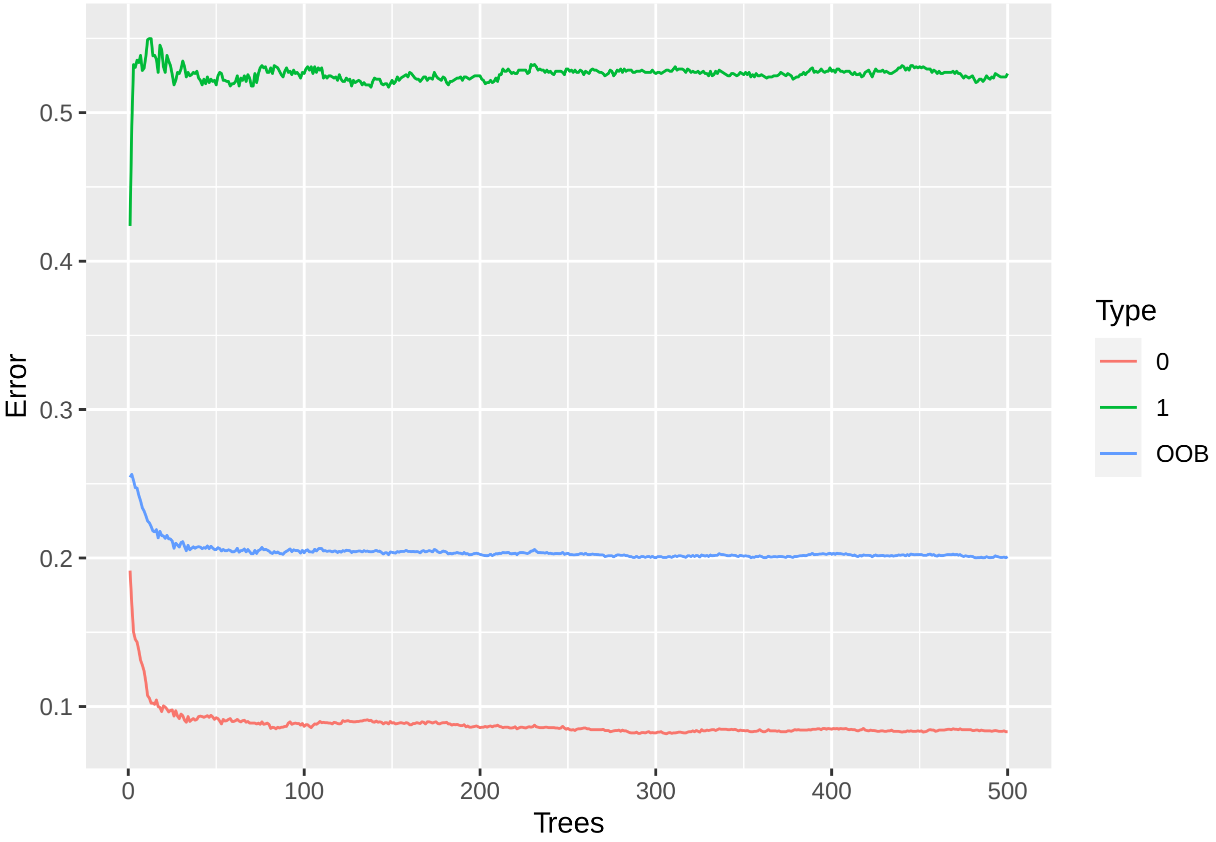
```
##
##     === AUCs ===
##
##      Model name Dataset ID Curve type       AUC
##    1         m1          1        ROC 0.8497658
##    2         m1          1        PRC 0.6668204
##
##
##     === Input data ===
##
##      Model name Dataset ID # of negatives # of positives
##    1         m1          1           1548            560
```

```
autoplot(precrec.rf)
```



### OOB error plot for optimised model

```
oob.error.data <- data.frame(Trees=rep(1:nrow(rfmodel$err.rate), times=3), Type=rep(c("OOB", "0", "1"),
ggplot(data=oob.error.data, aes(x=Trees, y=Error))+geom_line(aes(color=Type))
```



## Feature Importances across models

```
imp.logreg <- varImp(logreg, scale = FALSE)
imp.logreg
```

```
##                                  Overall
## genderMale                    0.59838111
## SeniorCitizen                 1.32992665
## PartnerYes                    0.45722380
## DependentsYes                 0.75218836
## tenure                        3.94587331
## PhoneServiceYes               0.03320498
## MultipleLinesYes              2.70322796
## InternetServiceFiber optic    1.99221781
## InternetServiceNo             1.61124858
## OnlineSecurityYes             0.69214973
## OnlineBackupYes               0.07558182
## DeviceProtectionYes           0.68767678
## TechSupportYes                0.45358284
## StreamingTVYes                1.57905858
## StreamingMoviesYes            1.42234902
## ContractOne year              4.16675335
## ContractTwo year              5.45076283
## PaperlessBillingYes           4.13144246
## PaymentMethodCredit card (automatic) 0.73908013
## PaymentMethodElectronic check        2.63620806
## PaymentMethodMailed check            0.89929640
## MonthlyCharges                       0.88813459
## MonthlyChargesIntervals20-40         0.63660211
## MonthlyChargesIntervals40-60         0.71202178
## MonthlyChargesIntervals60-80         0.01715294
## MonthlyChargesIntervals80-100        0.06531244
## MonthlyChargesIntervalsAbove 100     0.30200190
## TotalCharges                         3.72962419
## TotalChargesIntervals1000-2000       2.11906029
## TotalChargesIntervals2000-3000       4.14871423
## TotalChargesIntervals3000-4000       4.74988570
## TotalChargesIntervals4000-5000       4.80441973
## TotalChargesIntervals5000-6000       5.11032405
## TotalChargesIntervals6000-7000       5.28231786
## TotalChargesIntervals7000-8000       5.19335686
## TotalChargesIntervalsAbove 8000      4.28326648
```

```
imp.tree <- varImp(cart.opt, scale = FALSE)
imp.tree
```

```
##                            Overall
## Contract                291.352851
## DeviceProtection         18.250475
## InternetService          96.466156
## MonthlyCharges            7.280132
## MultipleLines             2.842111
## OnlineBackup             32.131150
## OnlineSecurity          343.114904
## PaperlessBilling          5.985914
## PaymentMethod           246.841296
## TechSupport             303.520728
## tenure                  390.958651
## TotalCharges            106.418423
## TotalChargesIntervals    58.089070
## gender                    0.000000
## SeniorCitizen             0.000000
## Partner                   0.000000
## Dependents                0.000000
## PhoneService              0.000000
## StreamingTV               0.000000
## StreamingMovies           0.000000
## MonthlyChargesIntervals   0.000000
```

```
imp.rfmodeloptim <- varImp(rfmodeloptim, scale = FALSE)
imp.rfmodeloptim
```

```
##                            Overall
## gender                   22.521053
## SeniorCitizen            20.729294
## Partner                  21.668583
## Dependents               20.240538
## tenure                  165.620070
## PhoneService              7.233984
## MultipleLines            25.219132
## InternetService          46.396192
## OnlineSecurity           59.639262
## OnlineBackup             37.534687
## DeviceProtection         33.560862
## TechSupport              54.440465
## StreamingTV              20.359073
## StreamingMovies          21.690099
## Contract                 93.283007
## PaperlessBilling         31.439824
## PaymentMethod            55.068175
## MonthlyCharges          136.733289
## MonthlyChargesIntervals  46.835585
## TotalCharges            161.657314
## TotalChargesIntervals    65.420257
```