Bank Churn Dataset Analysis

Import dependencies

```r
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
```

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

```
## corrplot 0.84 loaded
```

```r
library(gridExtra)
```

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

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

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

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

```r
library(MASS)
```

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

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

```r
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

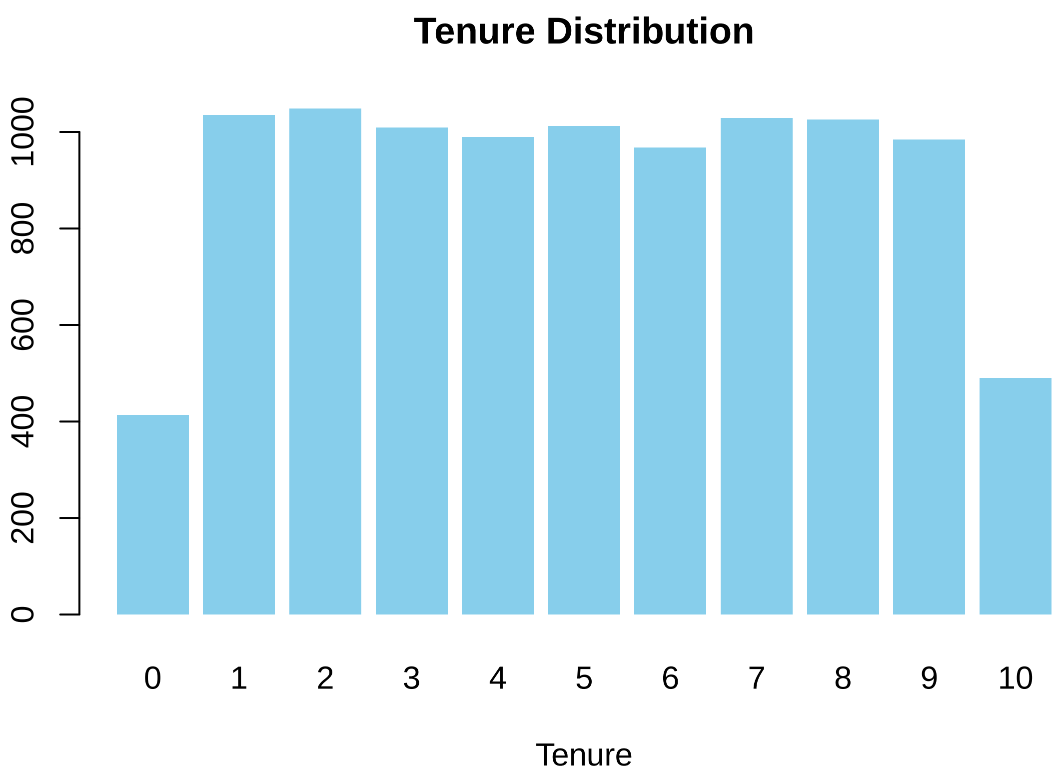
```r
bankChurn <- read.csv(file = 'Churn_Modelling.csv')
bankChurn = subset(bankChurn, select=-c(Surname, CustomerId, RowNumber, CreditScore, Geography, Gender,
names(bankChurn)[names(bankChurn) == "Exited"] <- "Churn"
bankChurn$Churn = ifelse(bankChurn$Churn == 1, "Yes", "No")
bankChurn$Churn = as.factor(bankChurn$Churn)
dim(bankChurn)
```

```
## [1] 10000     7
```

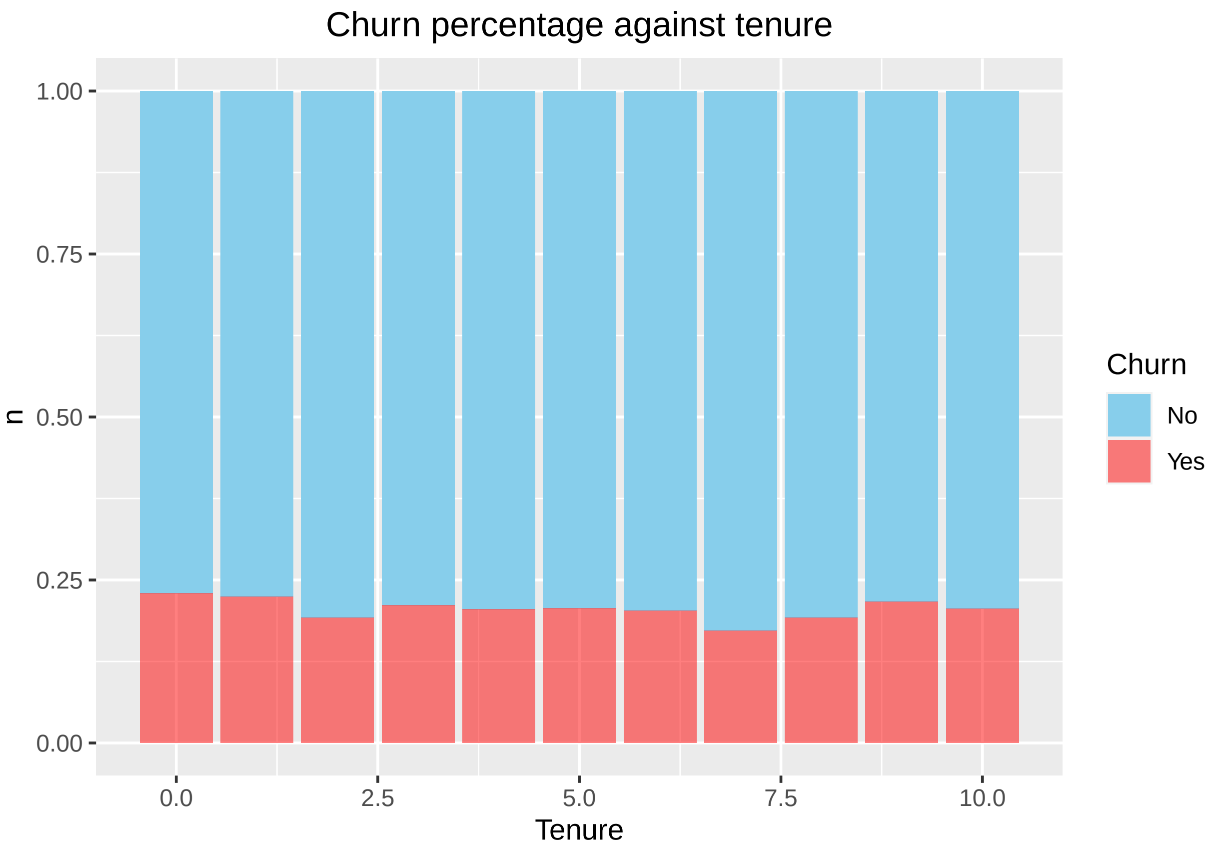
Exploratory Data Analysis

Tenure against Churn

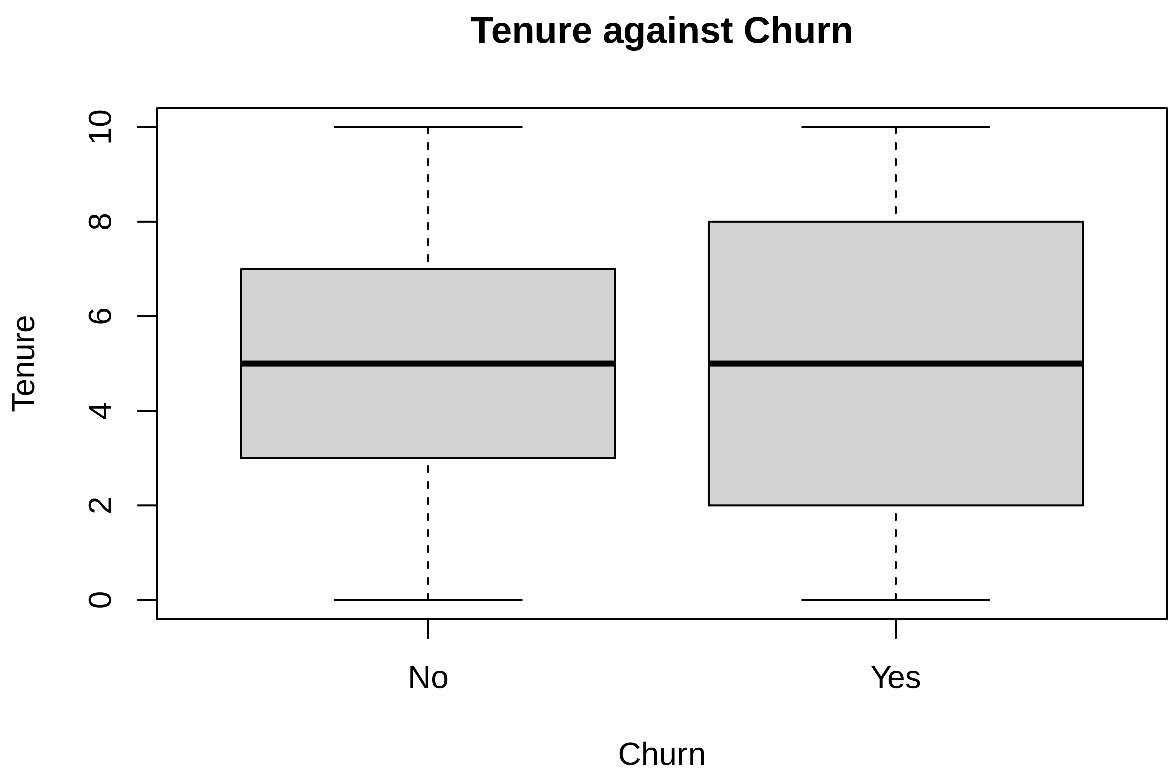
```r
tenureCounts <- table(bankChurn$Tenure)
barplot(
  tenureCounts,
  main="Tenure Distribution",
  xlab="Tenure",
  col="skyblue",
  border=F
  )
```




```r
# percentage Churn for each tenure
tenure <- bankChurn %>% count(Churn, Tenure)

ggplot(tenure, aes(fill=Churn, y=n, x=Tenure))+
  ggtitle("Churn percentage against tenure") +
  geom_bar(position="fill", stat="identity") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values=c("skyblue", scales::alpha("red", .5)))
```

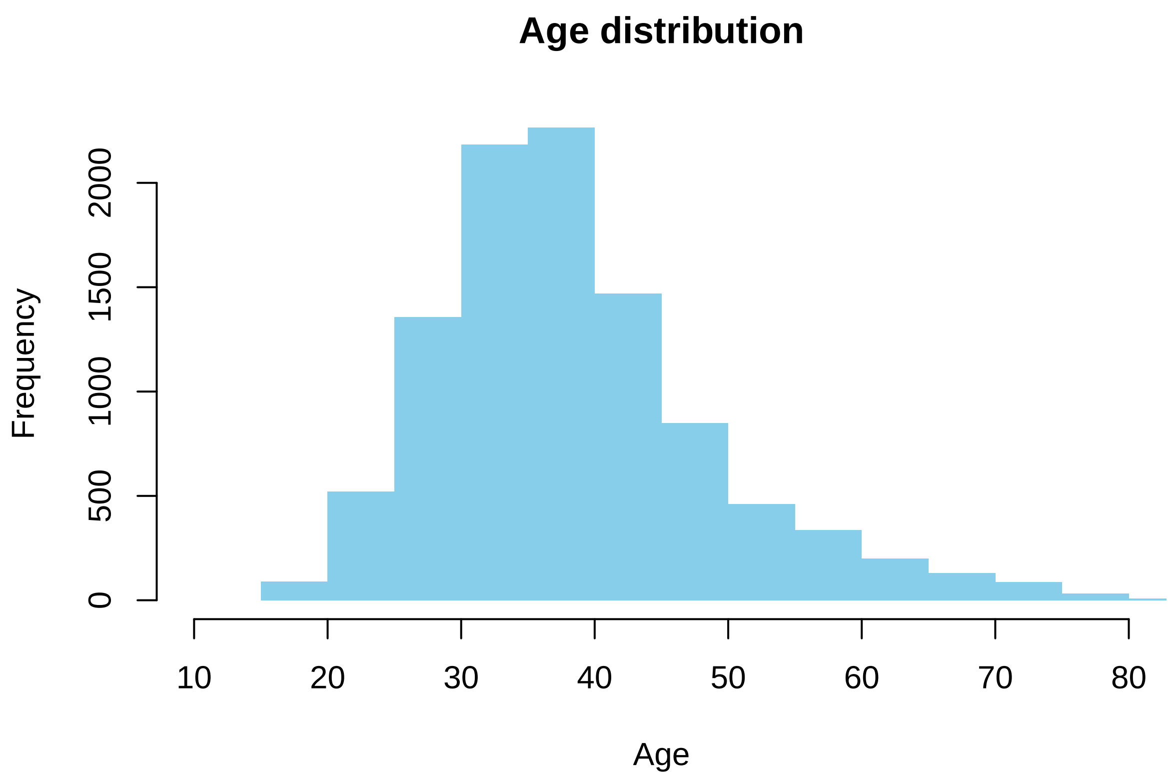



```
# boxplot for tenure against Churn
boxplot(
  Tenure ~ Churn,
  data=bankChurn,
  main="Tenure against Churn",
  xlab="Churn",
  ylab="Tenure"
  )
```

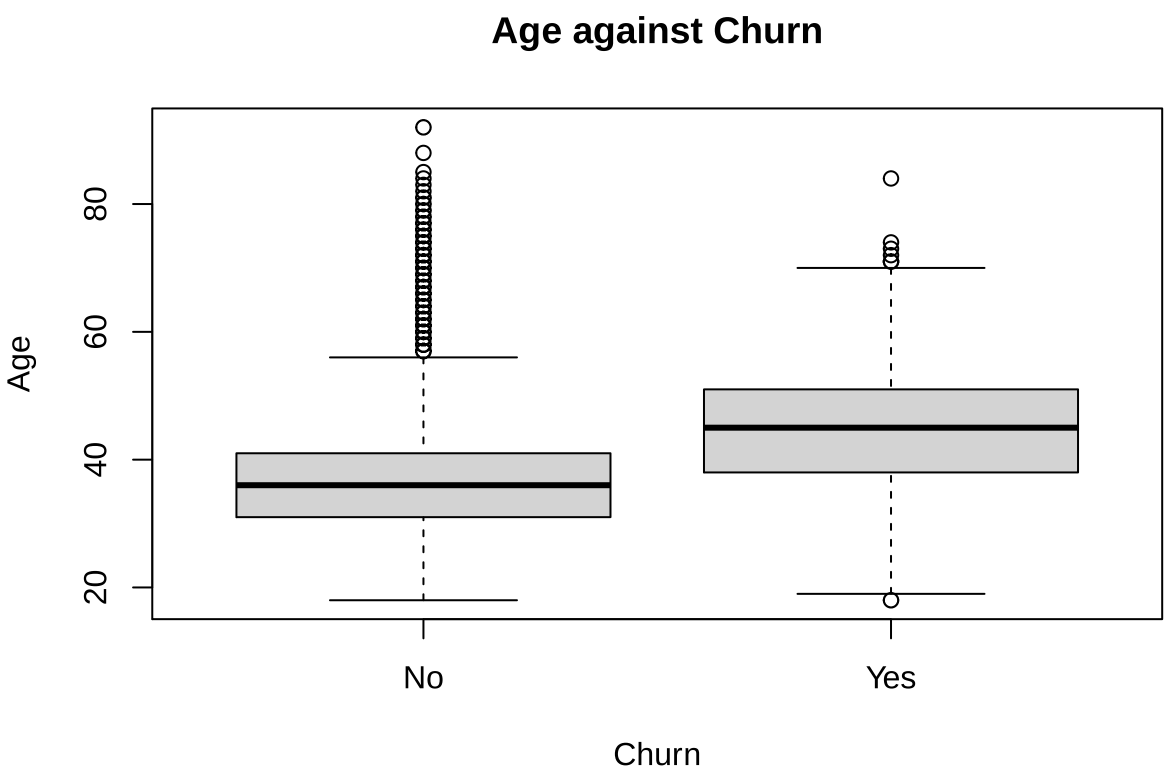


Age against Churn

```
# age distribution
hist(
  main="Age distribution",
  xlab="Age",
  bankChurn$Age,
  xlim=c(10,80),
  col='skyblue',
  border=F)
```

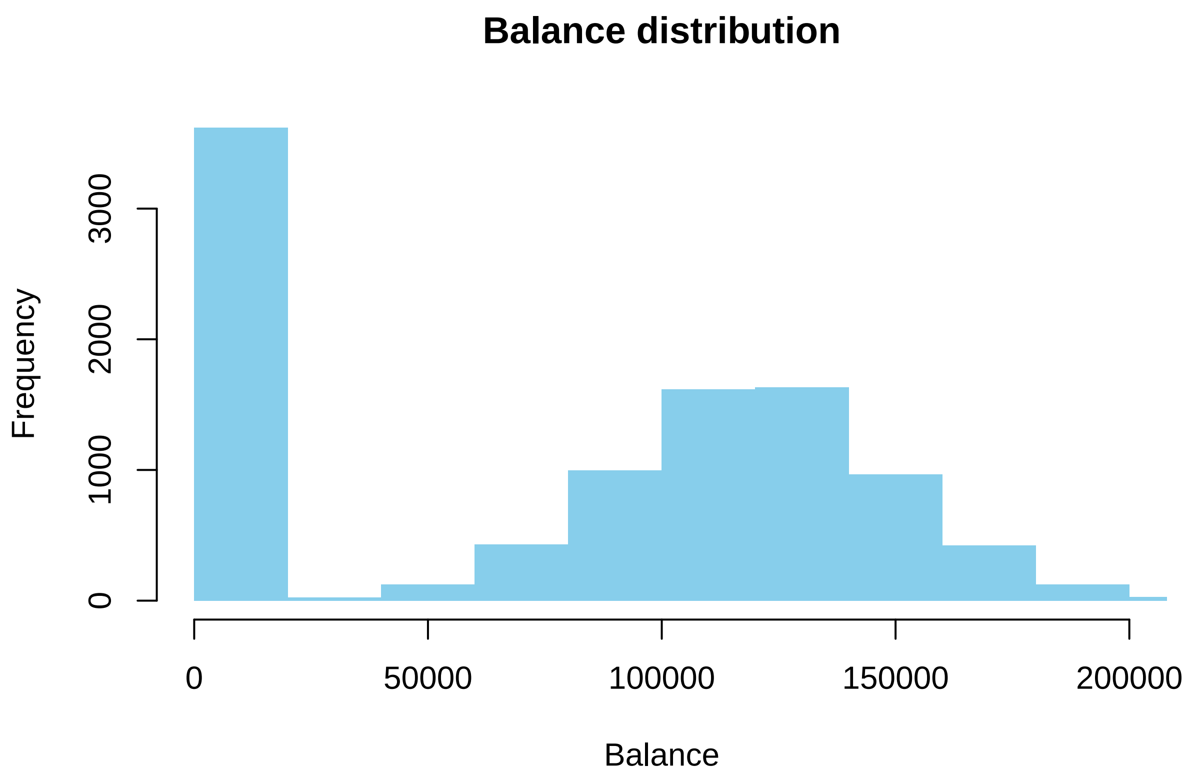



```
# boxplot for age against Churn
boxplot(
  Age ~ Churn,
  data=bankChurn,
  main="Age against Churn",
  xlab="Churn",
  ylab="Age"
  )
```

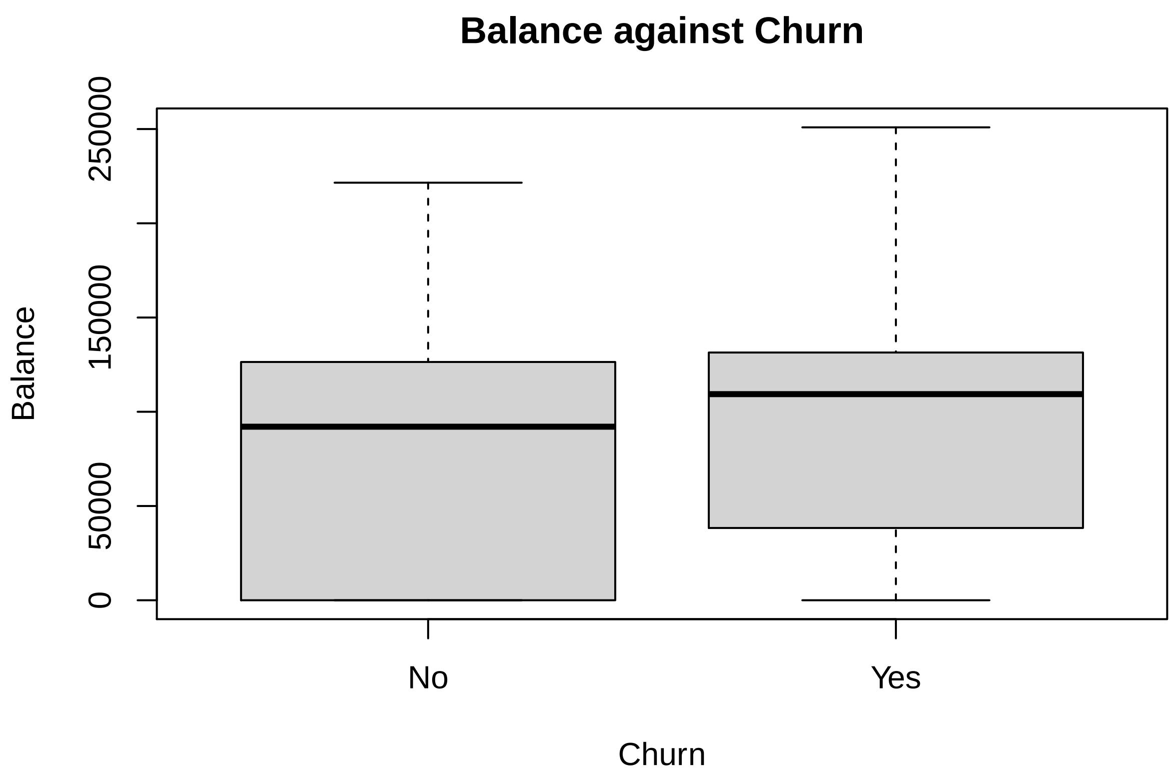


Balance against Churn

```
# balance distribution
hist(
  main="Balance distribution",
  xlab="Balance",
  bankChurn$Balance,
  xlim=c(0,200000),
  col='skyblue',
  border=F)
```

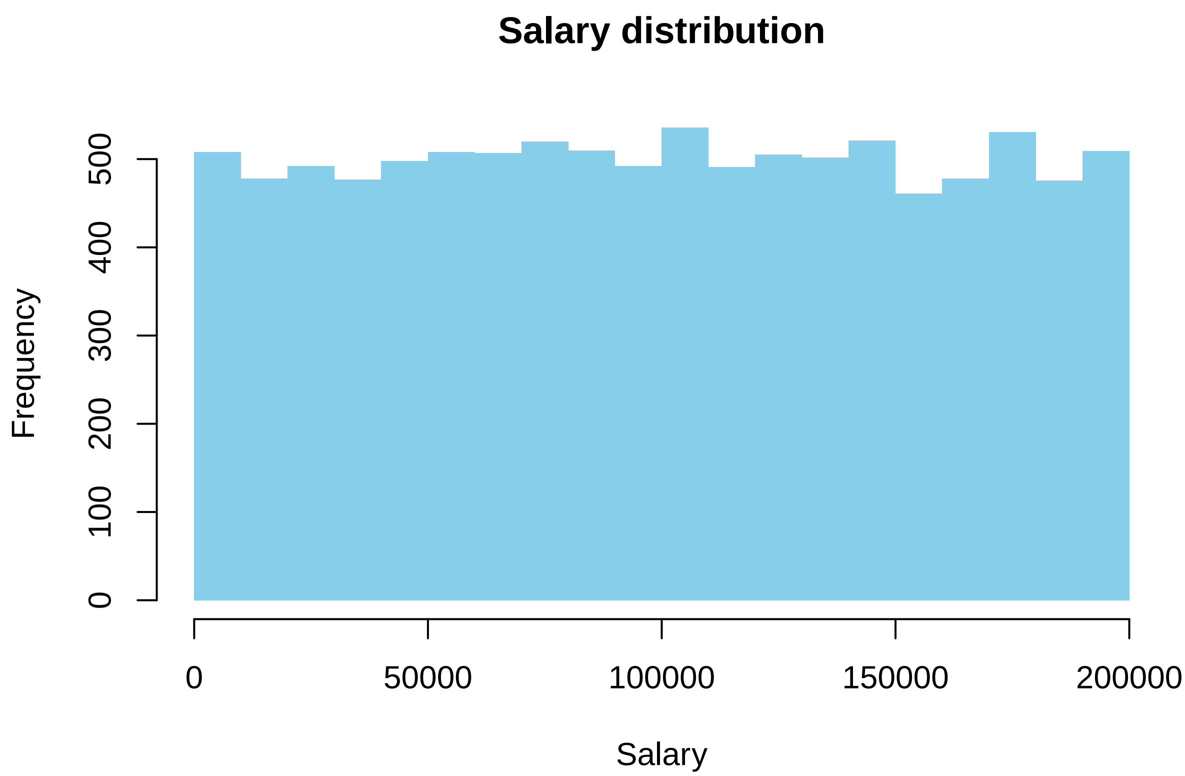



```r
# boxplot for balance against Churn
boxplot(
  Balance ~ Churn,
  data=bankChurn,
  main="Balance against Churn",
  xlab="Churn",
  ylab="Balance"
  )
```



Salary against Churn

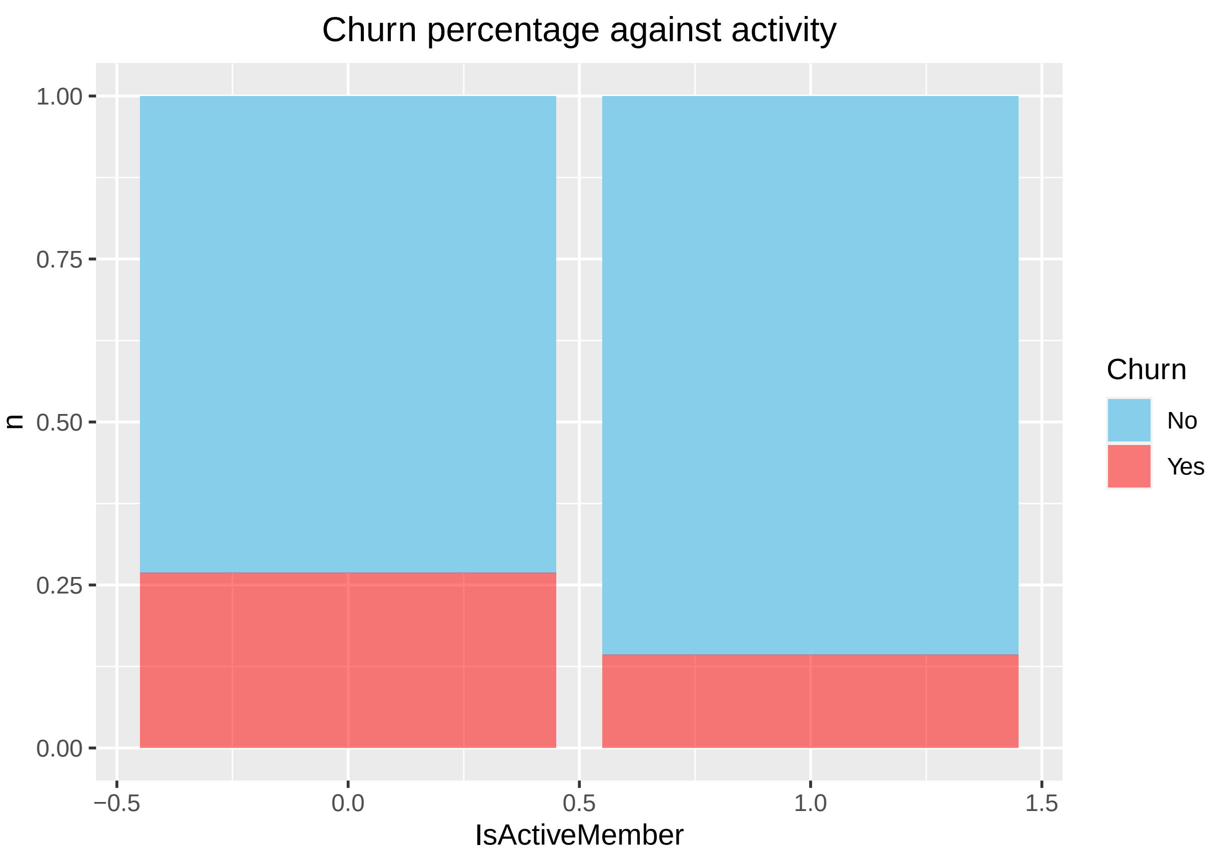
```r
# salary distribution
hist(
  main="Salary distribution",
  xlab="Salary",
  bankChurn$EstimatedSalary,
  xlim=c(0,200000),
  col='skyblue',
  border=F)
```




```r
# boxplot for salary against Churn
boxplot(
  EstimatedSalary ~ Churn,
  data=bankChurn,
  main="Salary against Churn",
  xlab="Churn",
  ylab="Salary"
  )
```

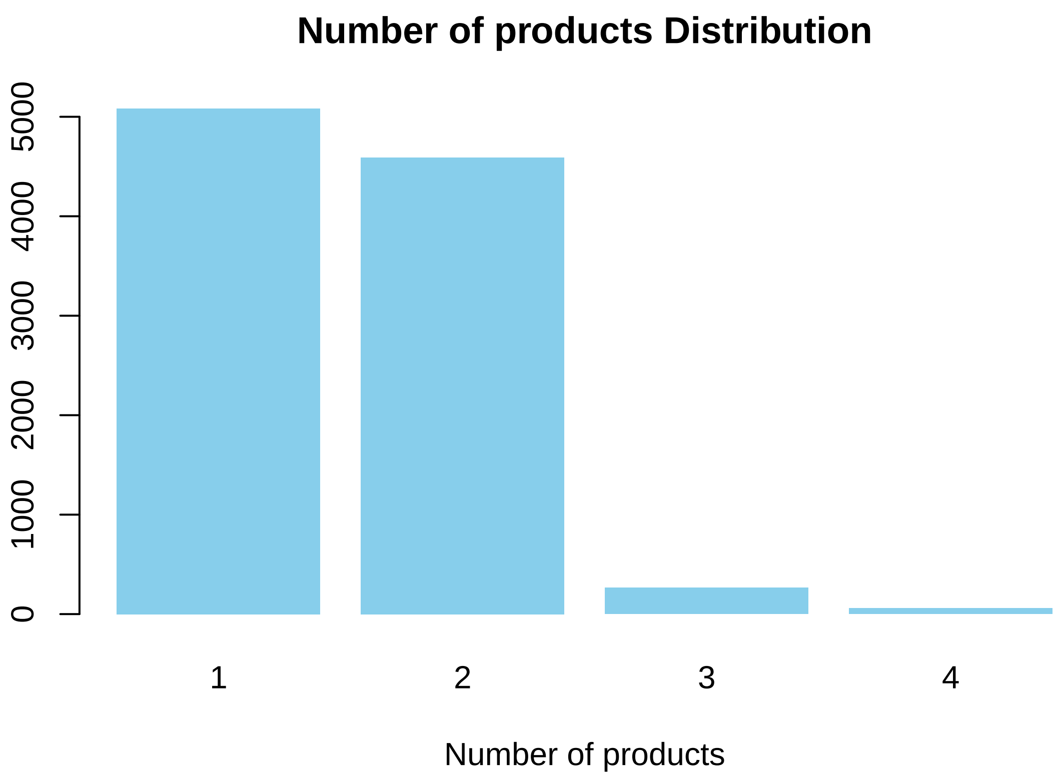
Activity against Churn

```r
# percentage Churn against is_active
isActive <- bankChurn %>% count(Churn, IsActiveMember)
ggplot(isActive, aes(fill=Churn, y=n, x=IsActiveMember))+
  ggtitle("Churn percentage against activity")+
  geom_bar(position="fill", stat="identity")+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_fill_manual(values=c("skyblue", scales::alpha("red", .5)))
```

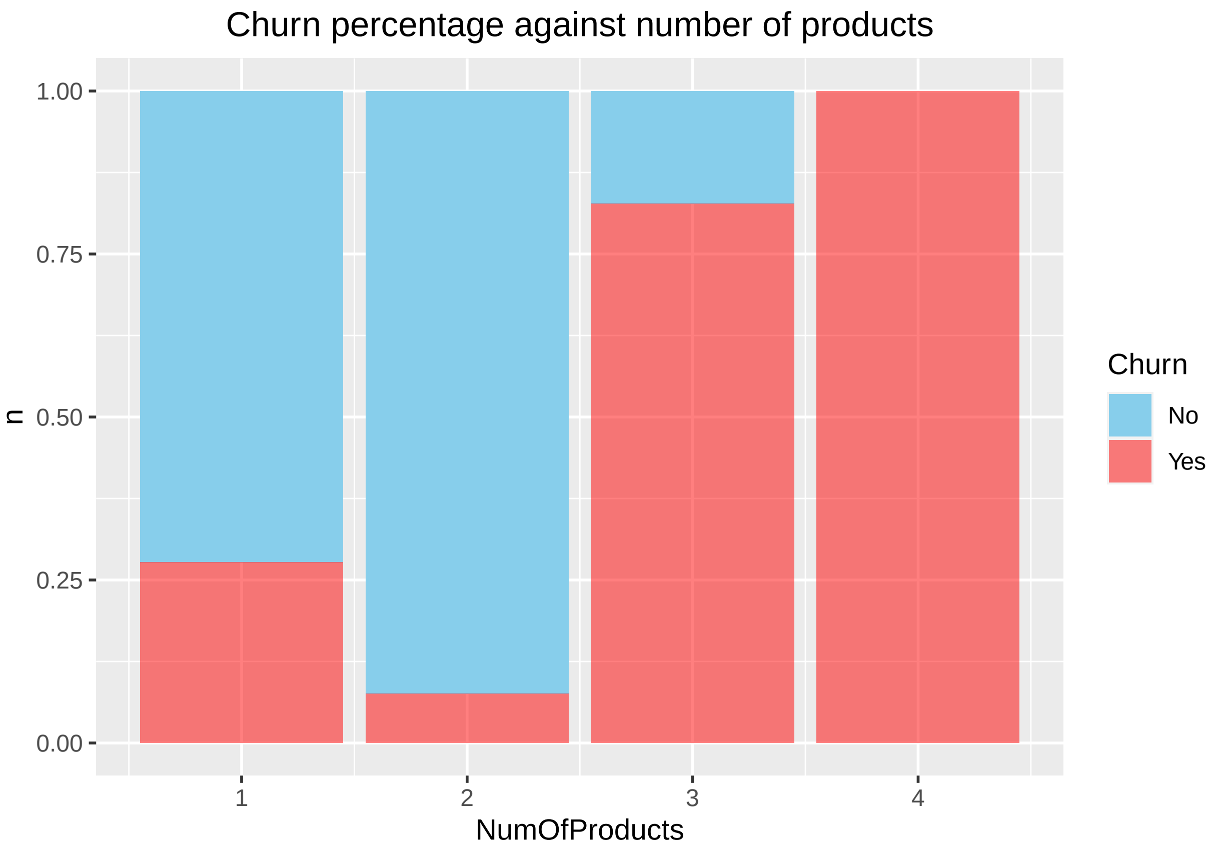


Number of Products against Churn

```
numProdCounts <- table(bankChurn$NumOfProducts)
barplot(
  numProdCounts,
  main="Number of products Distribution",
  xlab="Number of products",
  col="skyblue",
  border=F
  )
```




```
# percentage Churn against number of products
numProducts <- bankChurn %>% count(Churn, NumOfProducts)

ggplot(numProducts, aes(fill=Churn, y=n, x=NumOfProducts))+
  ggtitle("Churn percentage against number of products")+
  geom_bar(position="fill", stat="identity")+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_fill_manual(values=c("skyblue", scales::alpha("red", .5)))
```



Training

Train-test-split

```
# train test split
idx = createDataPartition(bankChurn$Churn, p=0.7, list=FALSE)
set.seed(42)
train = bankChurn[idx,]
test = bankChurn[-idx,]
train$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] 7001    7
```

```
## [1] 2999    7
```

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
## -2.0584  -0.6799  -0.4786  -0.2979   2.8628
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.175e+00  1.865e-01 -22.383   <2e-16 ***
## Age              7.130e-02  3.003e-03  23.744   <2e-16 ***
## Tenure          -1.281e-02  1.098e-02  -1.167   0.2434
## Balance          4.804e-06  5.466e-07   8.789   <2e-16 ***
## NumOfProducts   -2.372e-02  5.528e-02  -0.429   0.6679
## IsActiveMember  -1.107e+00  6.793e-02 -16.292   <2e-16 ***
## EstimatedSalary  9.798e-07  5.571e-07   1.759   0.0786 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7077.6  on 7000  degrees of freedom
## Residual deviance: 6185.5  on 6994  degrees of freedom
## AIC: 6199.5
##
## Number of Fisher Scoring iterations: 5
```

Feature importance using deviance

```r
# 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                             7000     7077.6
## Age              1   508.20      6999     6569.4  < 2e-16 ***
## Tenure           1     0.45      6998     6568.9  0.50074
## Balance          1    93.12      6997     6475.8  < 2e-16 ***
## NumOfProducts    1     0.50      6996     6475.3  0.47871
## IsActiveMember   1   286.75      6995     6188.6  < 2e-16 ***
## EstimatedSalary  1     3.10      6994     6185.5  0.07849 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Accuracy

```r
# evaluating logistic regression model against test data

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

```
## [1] "Logistic Regression Accuracy 0.811270423474492"
```

Confusion Matrix

```r
print("Confusion Matrix for Logistic Regression")
```

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

```r
table(Predicted = logreg.pred, Actual = test$Churn)
```

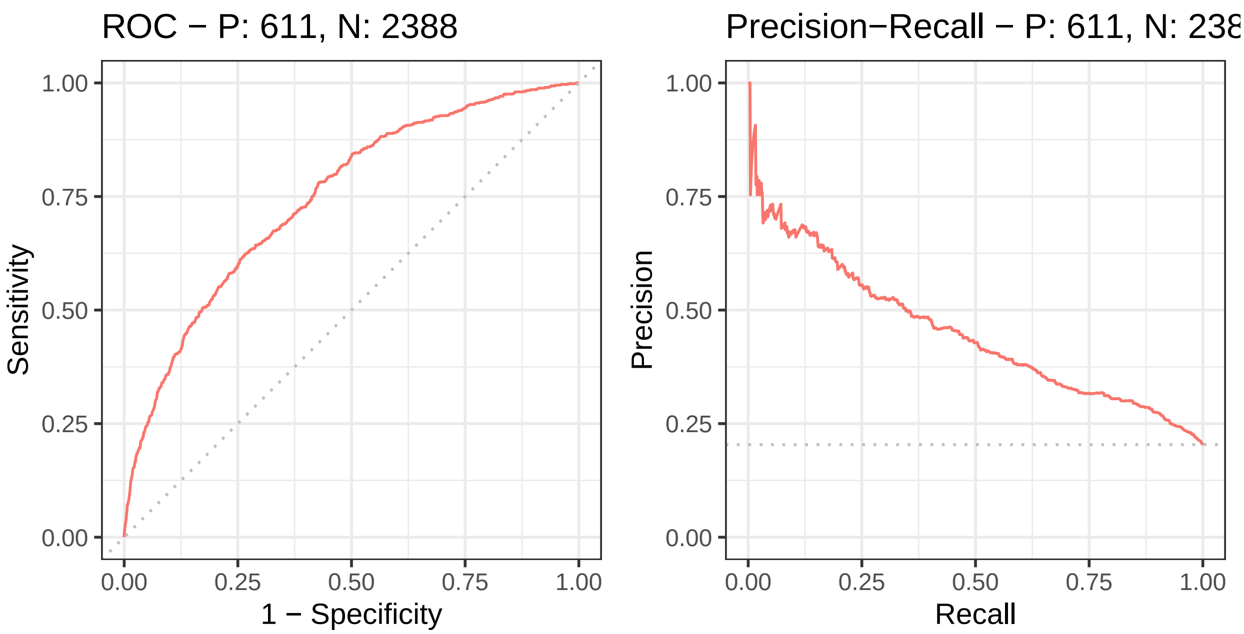
```
##          Actual
## Predicted    0    1
##         0 2332  510
##         1   56  101
```

ROC, PRC curves

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

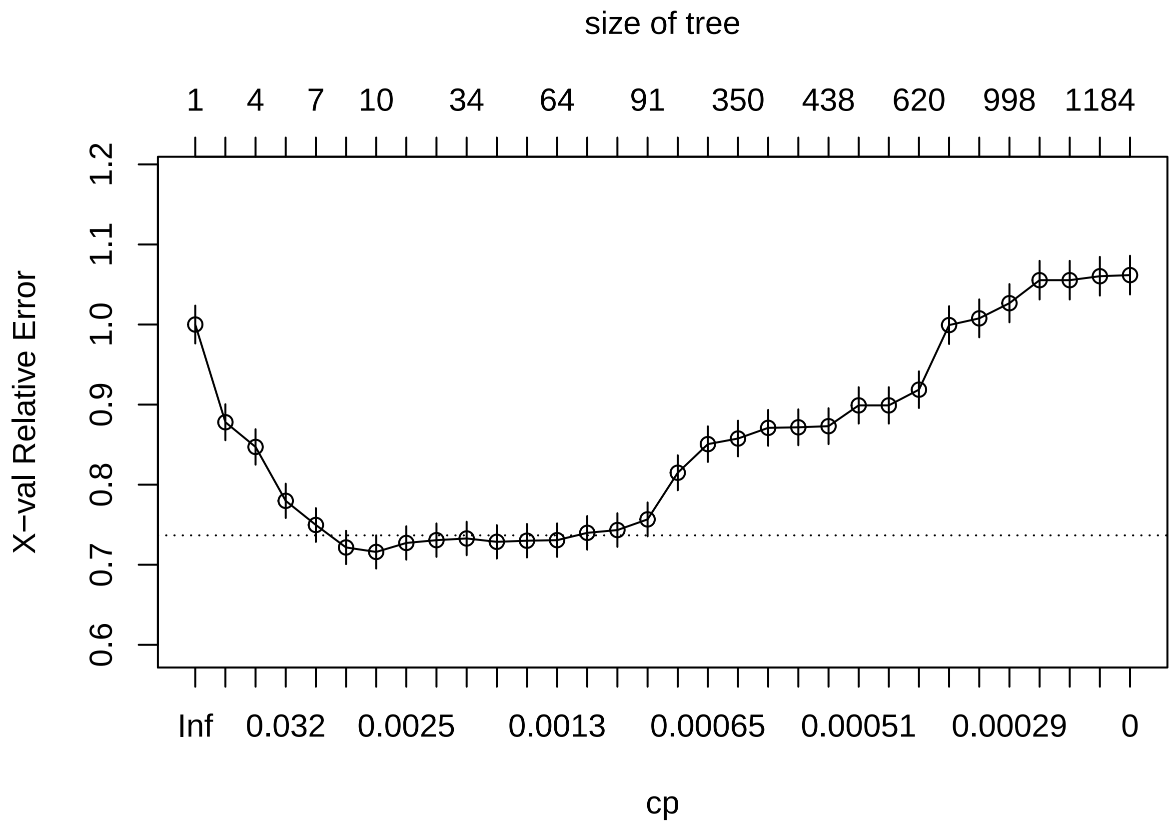
```
## 
##     === AUCs ===
## 
##      Model name Dataset ID Curve type       AUC
##    1         m1          1        ROC 0.7466424
##    2         m1          1        PRC 0.4512625
## 
## 
##     === Input data ===
## 
##      Model name Dataset ID # of negatives # of positives
##    1         m1          1           2388            611
```

```r
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)
```

Feature importance

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

```
##            Age   NumOfProducts  IsActiveMember         Balance EstimatedSalary
##     316.609073      205.676795      109.882162        6.645536        2.057897
##         Tenure
##       1.955634
```

Accuracy

```
tree.pred <- predict(cart.opt, test, type="class")
tree.pred.scores <- predict(cart.opt, test, type="prob")
```

```r
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 0.853951317105702"
```

Confusion Matrix

```r
# 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)
```

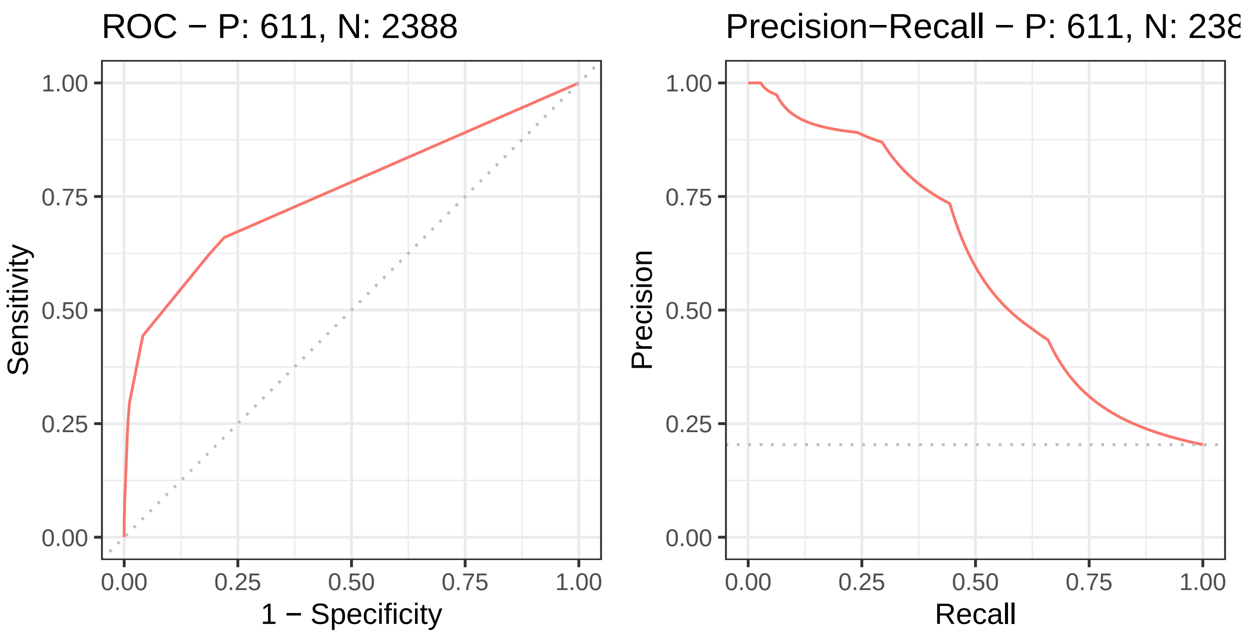
```
## [1] "Confusion Matrix for Decision Tree"

##          Actual
## Predicted    0    1
##         0 2290  340
##         1   98  271
```

ROC, PRC curves

```r
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 0.7593251
##    2         m1          1        PRC 0.6006364
##
##
##     === Input data ===
##
##      Model name Dataset ID # of negatives # of positives
##    1         m1          1           2388            611
```

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



Random Forest

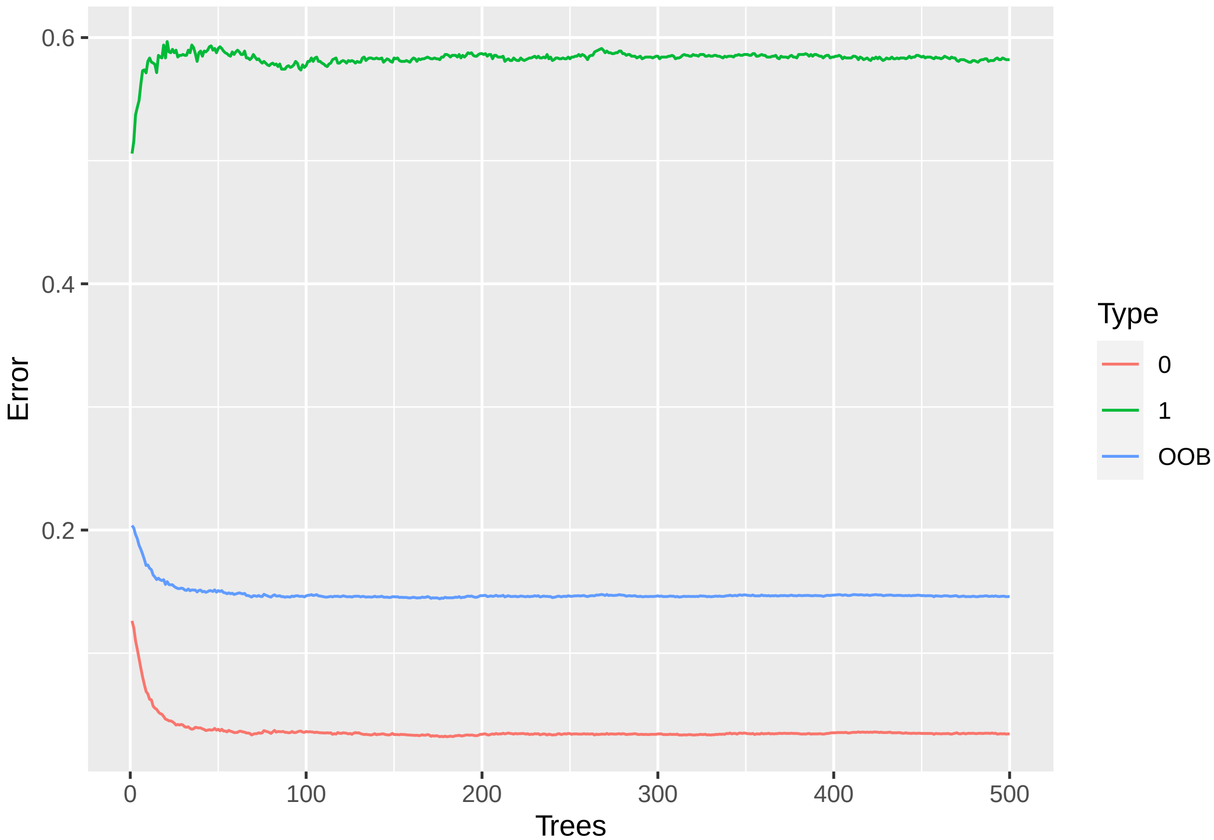
```r
rf <- randomForest(Churn ~ ., data = train, proximity = TRUE, type='classification')
```

```r
rf
```

```
##
## Call:
##  randomForest(formula = Churn ~ ., data = train, proximity = TRUE,      type = "classification")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 14.6%
## Confusion matrix:
##      0   1 class.error
## 0 5383 192  0.03443946
## 1  830 596  0.58204769
```

OOB error plot

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

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



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 = train, 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]
}
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
oob.values
```

```
##  [1] 0.1529781 0.1474075 0.1506928 0.1538352 0.1545494 0.1565491 0.1561206
##  [8] 0.1556920 0.1569776 0.1562634
```

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

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

```r
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: 14.96%
## Confusion matrix:
##      0   1 class.error
## 0 5374 201  0.03605381
## 1  846 580  0.59326788
```

```r
# error rate reduced
```

Accuracy

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

```
## [1] "Random Forest Accuracy 0.856952317439146"
```

Confusion Matrix

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

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

```
##          Actual
## Predicted    0    1
##         0 2312  353
##         1   76  258
```

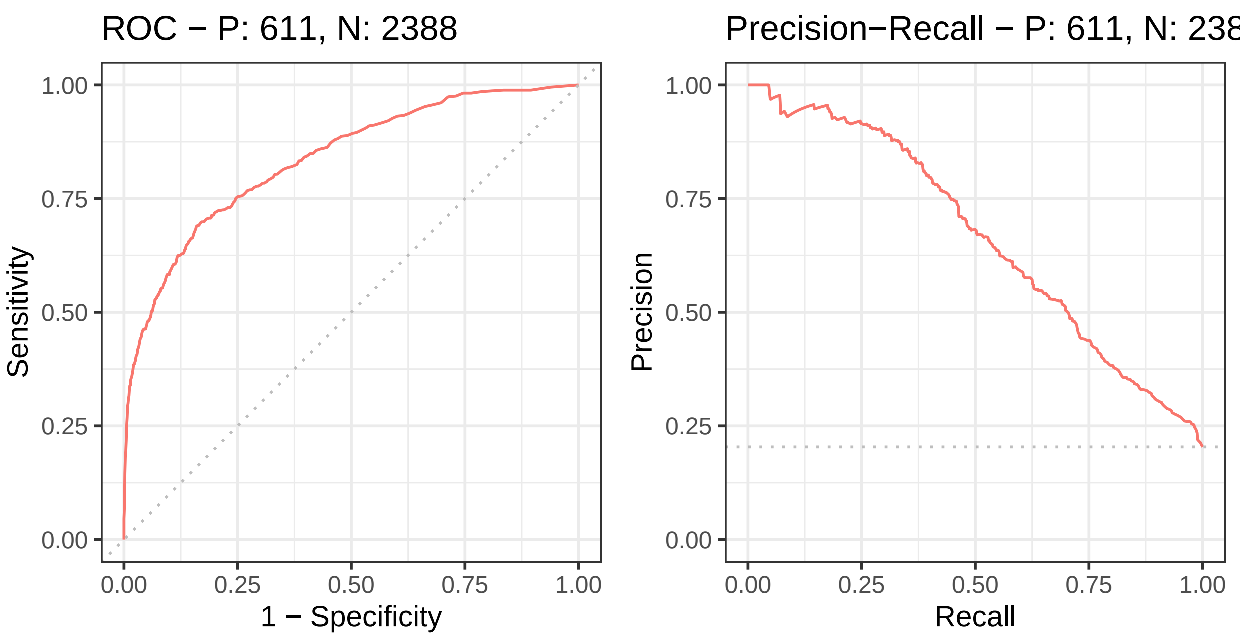
ROC, PRC curves

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

```
##
##     === AUCs ===
##
##      Model name Dataset ID Curve type       AUC
##    1         m1          1        ROC 0.8363438
##    2         m1          1        PRC 0.6653247
##
##
##     === Input data ===
```

```
##
##      Model name Dataset ID # of negatives # of positives
##    1         m1          1           2388            611
```

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



Feature Importances for all models

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

```
##                    Overall
## Age             23.7438976
## Tenure           1.1665774
## Balance          8.7885200
## NumOfProducts    0.4290583
## IsActiveMember  16.2923385
## EstimatedSalary  1.7587539
```

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

```
##                  Overall
## Age             386.9886
## Balance         108.4031
## EstimatedSalary  15.6846
## IsActiveMember  179.9219
## NumOfProducts   503.5642
## Tenure           11.8388
```

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

```
##                   Overall
## Age             517.19454
```

```
## Tenure          161.21682
## Balance         322.79030
## NumOfProducts   295.44449
## IsActiveMember   90.41581
## EstimatedSalary 344.75715
```