Ecommerce Customer Churn Analysis & Prediction

Team 1: Chang-Han Chen/ Jingjing Ling/ Yuanming Zhang/ Linghan Leng/ Kangjing Shi / Zuoyin Li

2/27/2021

Packages

```
# Install Packages
# install.packages(c("rpart.plot", "rpart"))

# Load Packages
library(data.table)
library(ggplot2)
library(ggthemes)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1
```

```
library('fastDummies')
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:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
theme_set(theme_bw())
```

Read the file

```
dd <- fread("D:/Boston University Courses/BA 810 Class-Supervised Machine Learning/Team Project/E Comme
str(dd)
```

```
## Classes 'data.table' and 'data.frame':   5630 obs. of  20 variables:
##  $ CustomerID                 : int  50001 50002 50003 50004 50005 50006 50007 50008 50009 50010 ...
##  $ Churn                      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Tenure                     : int  4 NA NA 0 0 0 NA NA 13 NA ...
##  $ PreferredLoginDevice       : chr  "Mobile Phone" "Phone" "Phone" "Phone" ...
##  $ CityTier                   : int  3 1 1 3 1 1 3 1 3 1 ...
##  $ WarehouseToHome            : int  6 8 30 15 12 22 11 6 9 31 ...
##  $ PreferredPaymentMode       : chr  "Debit Card" "UPI" "Debit Card" "Debit Card" ...
##  $ Gender                     : chr  "Female" "Male" "Male" "Male" ...
##  $ HourSpendOnApp             : int  3 3 2 2 NA 3 2 3 NA 2 ...
##  $ NumberOfDeviceRegistered   : int  3 4 4 4 3 5 3 3 4 5 ...
##  $ PreferedOrderCat           : chr  "Laptop & Accessory" "Mobile" "Mobile" "Laptop & Accessory" ...
##  $ SatisfactionScore          : int  2 3 3 5 5 5 2 2 3 3 ...
##  $ MaritalStatus              : chr  "Single" "Single" "Single" "Single" ...
##  $ NumberOfAddress            : int  9 7 6 8 3 2 4 3 2 2 ...
##  $ Complain                   : int  1 1 1 0 0 1 0 1 1 0 ...
##  $ OrderAmountHikeFromlastYear: int  11 15 14 23 11 22 14 16 14 12 ...
##  $ CouponUsed                 : int  1 0 0 0 1 4 0 2 0 1 ...
##  $ OrderCount                 : int  1 1 1 1 1 6 1 2 1 1 ...
##  $ DaySinceLastOrder          : int  5 0 3 3 3 7 0 0 2 1 ...
##  $ CashbackAmount             : int  160 121 120 134 130 139 121 123 127 123 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
head(dd)
```

```
##    CustomerID Churn Tenure PreferredLoginDevice CityTier WarehouseToHome
## 1:      50001     1      4         Mobile Phone        3               6
## 2:      50002     1     NA                Phone        1               8
## 3:      50003     1     NA                Phone        1              30
## 4:      50004     1      0                Phone        3              15
## 5:      50005     1      0                Phone        1              12
## 6:      50006     1      0             Computer        1              22
##    PreferredPaymentMode Gender HourSpendOnApp NumberOfDeviceRegistered
## 1:           Debit Card Female              3                        3
## 2:                  UPI   Male              3                        4
## 3:           Debit Card   Male              2                        4
## 4:           Debit Card   Male              2                        4
## 5:                   CC   Male             NA                        3
## 6:           Debit Card Female              3                        5
##      PreferedOrderCat SatisfactionScore MaritalStatus NumberOfAddress Complain
## 1: Laptop & Accessory                 2        Single               9        1
## 2:             Mobile                 3        Single               7        1
## 3:             Mobile                 3        Single               6        1
```

```
## 4: Laptop & Accessory                 5        Single                 8         0
## 5:             Mobile                 5        Single                 3         0
## 6:       Mobile Phone                 5        Single                 2         1
##    OrderAmountHikeFromlastYear CouponUsed OrderCount DaySinceLastOrder
## 1:                          11          1          1                 5
## 2:                          15          0          1                 0
## 3:                          14          0          1                 3
## 4:                          23          0          1                 3
## 5:                          11          1          1                 3
## 6:                          22          4          6                 7
##    CashbackAmount
## 1:            160
## 2:            121
## 3:            120
## 4:            134
## 5:            130
## 6:            139
```

Data Cleaning

```r
 # delete the irrelevant column
dd[,CustomerID:= NULL]

# check columns with null values
colSums(is.na(dd))
```

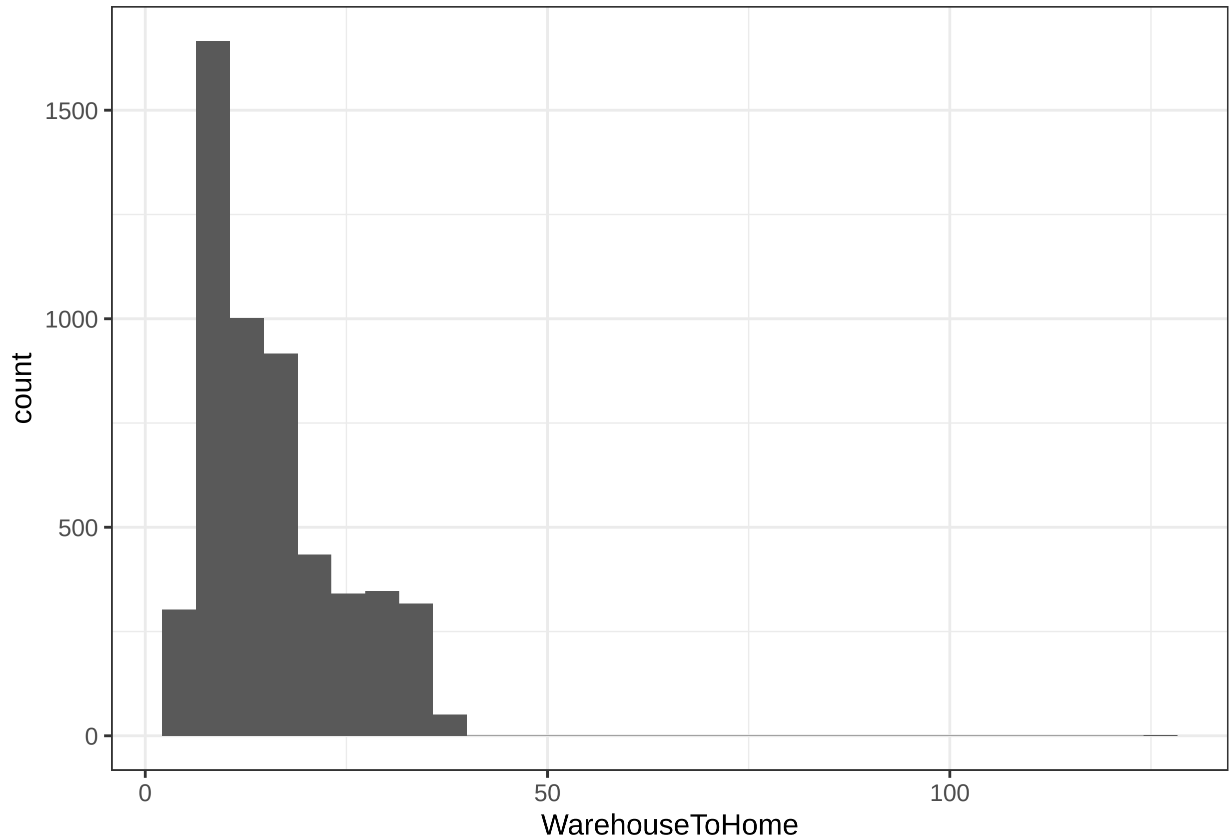
```
##                       Churn                      Tenure
##                           0                         264
##         PreferredLoginDevice                    CityTier
##                           0                           0
##              WarehouseToHome        PreferredPaymentMode
##                         251                           0
##                       Gender               HourSpendOnApp
##                           0                         255
##    NumberOfDeviceRegistered            PreferedOrderCat
##                           0                           0
##            SatisfactionScore               MaritalStatus
##                           0                           0
##              NumberOfAddress                    Complain
##                           0                           0
## OrderAmountHikeFromlastYear                  CouponUsed
##                         265                         256
##                   OrderCount           DaySinceLastOrder
##                         258                         307
##               CashbackAmount
##                           0
```

Before handle null values, see data distribution to determine which value to fill in

```r
ggplot(dd, aes(WarehouseToHome)) + geom_histogram()
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

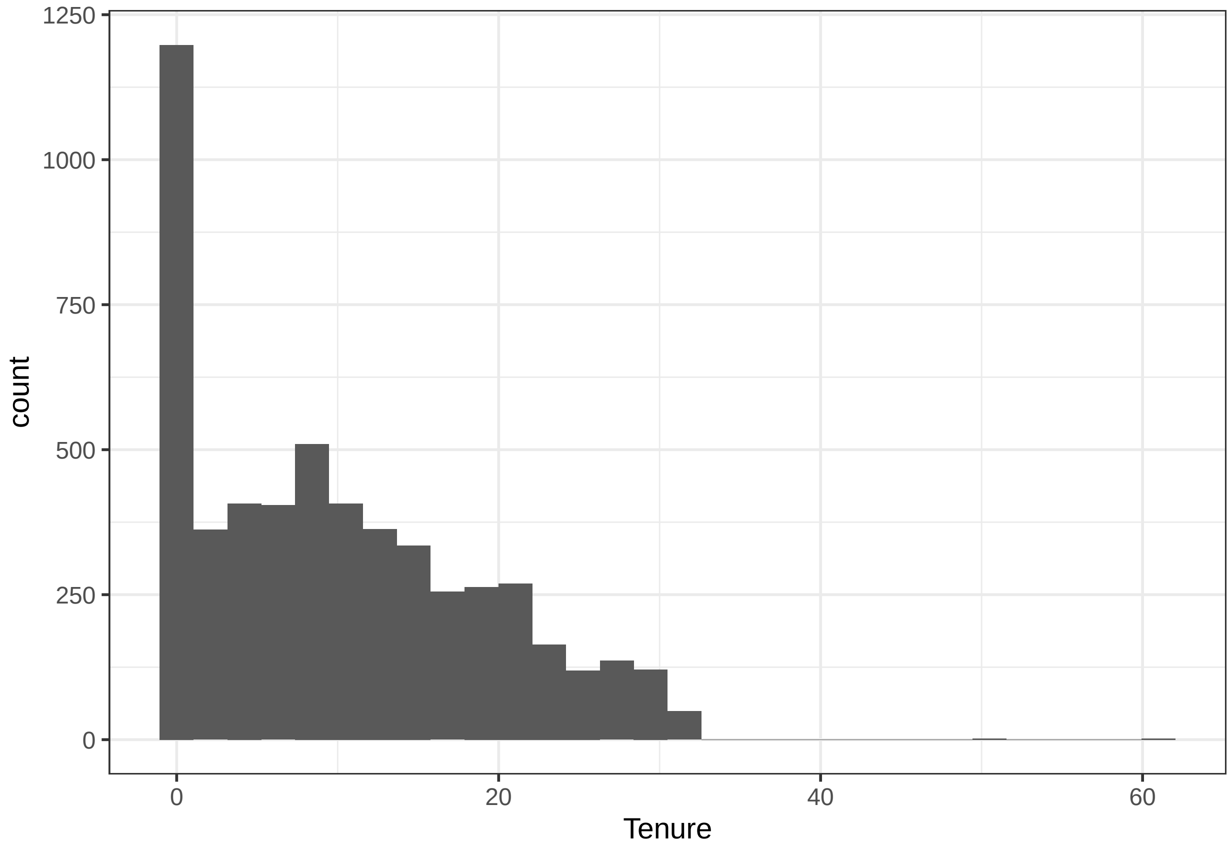
```
## Warning: Removed 251 rows containing non-finite values (stat_bin).
```




```
ggplot(dd, aes(Tenure)) + geom_histogram()
```

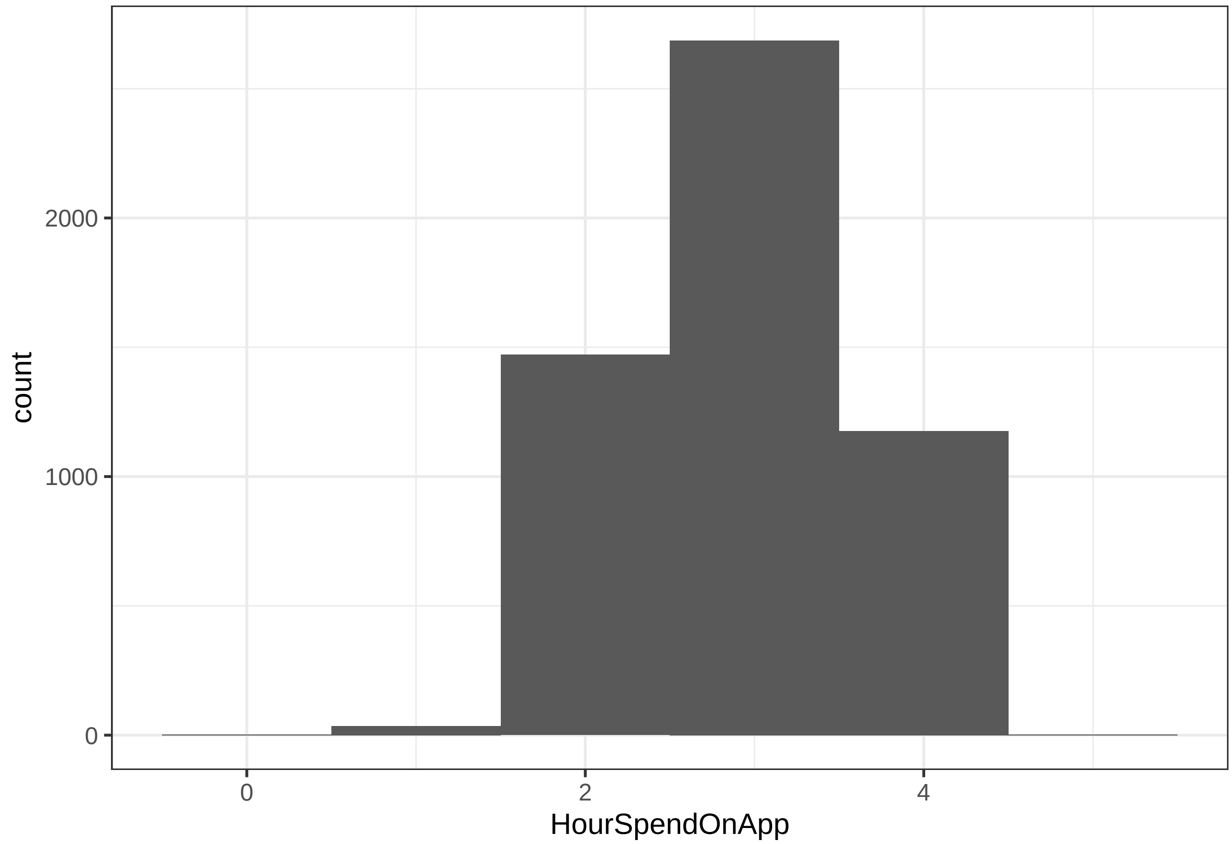
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 264 rows containing non-finite values (stat_bin).
```



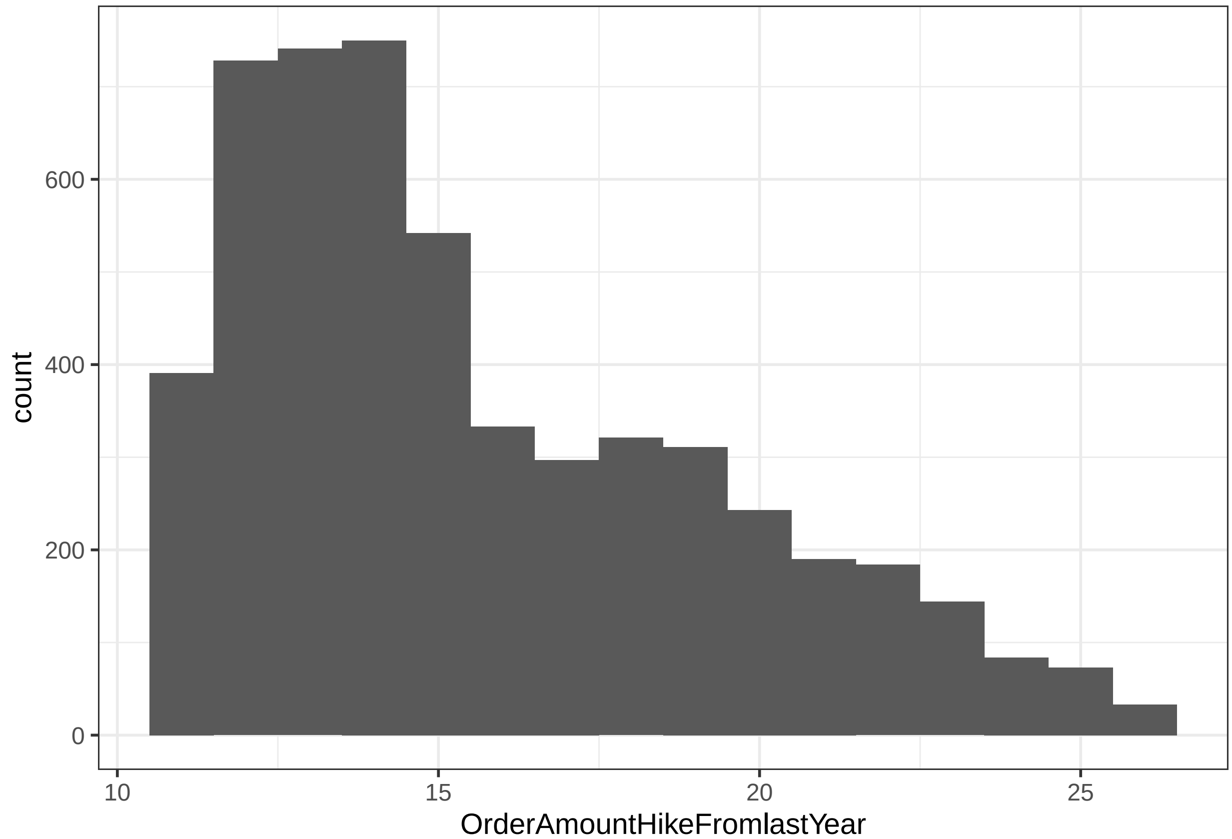

```
ggplot(dd, aes(HourSpendOnApp)) + geom_histogram(binwidth = 1)
```

```
## Warning: Removed 255 rows containing non-finite values (stat_bin).
```



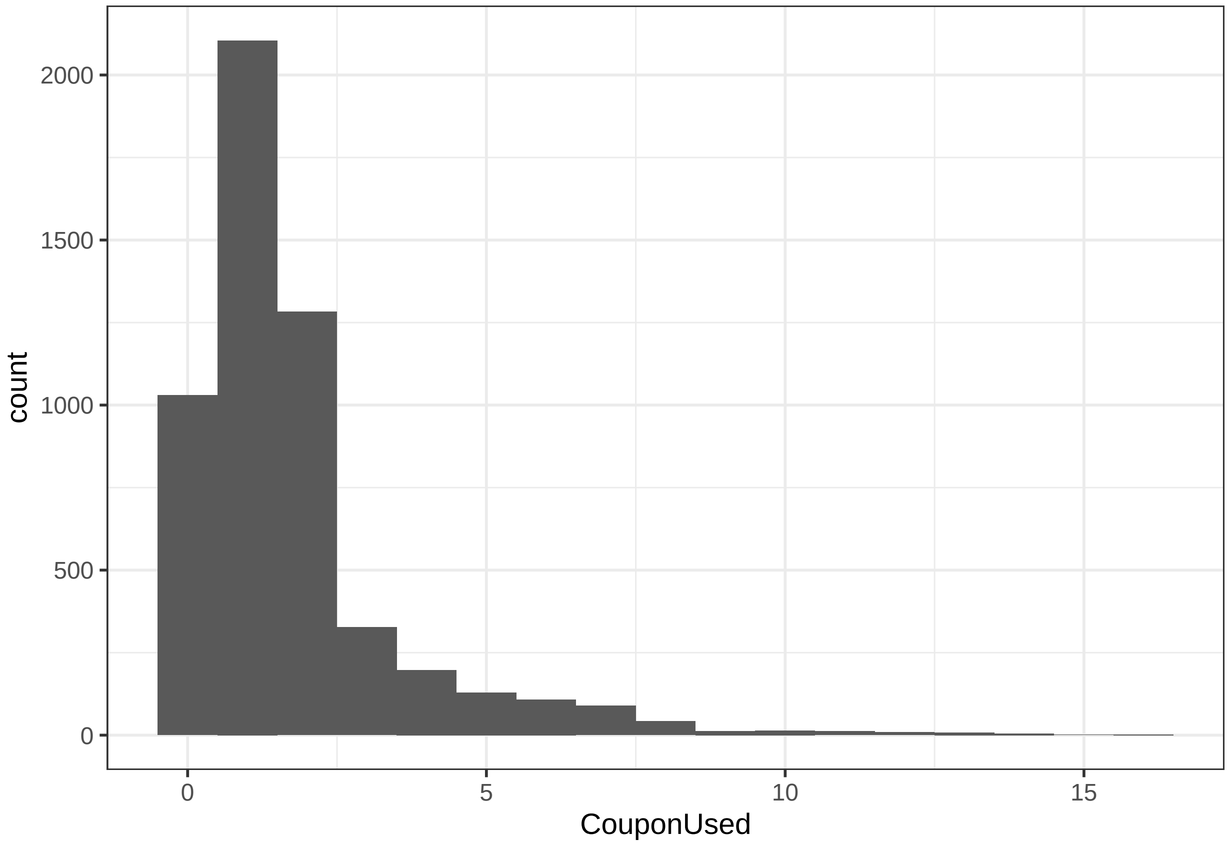

```
ggplot(dd, aes(OrderAmountHikeFromlastYear)) + geom_histogram(binwidth = 1)
```

```
## Warning: Removed 265 rows containing non-finite values (stat_bin).
```



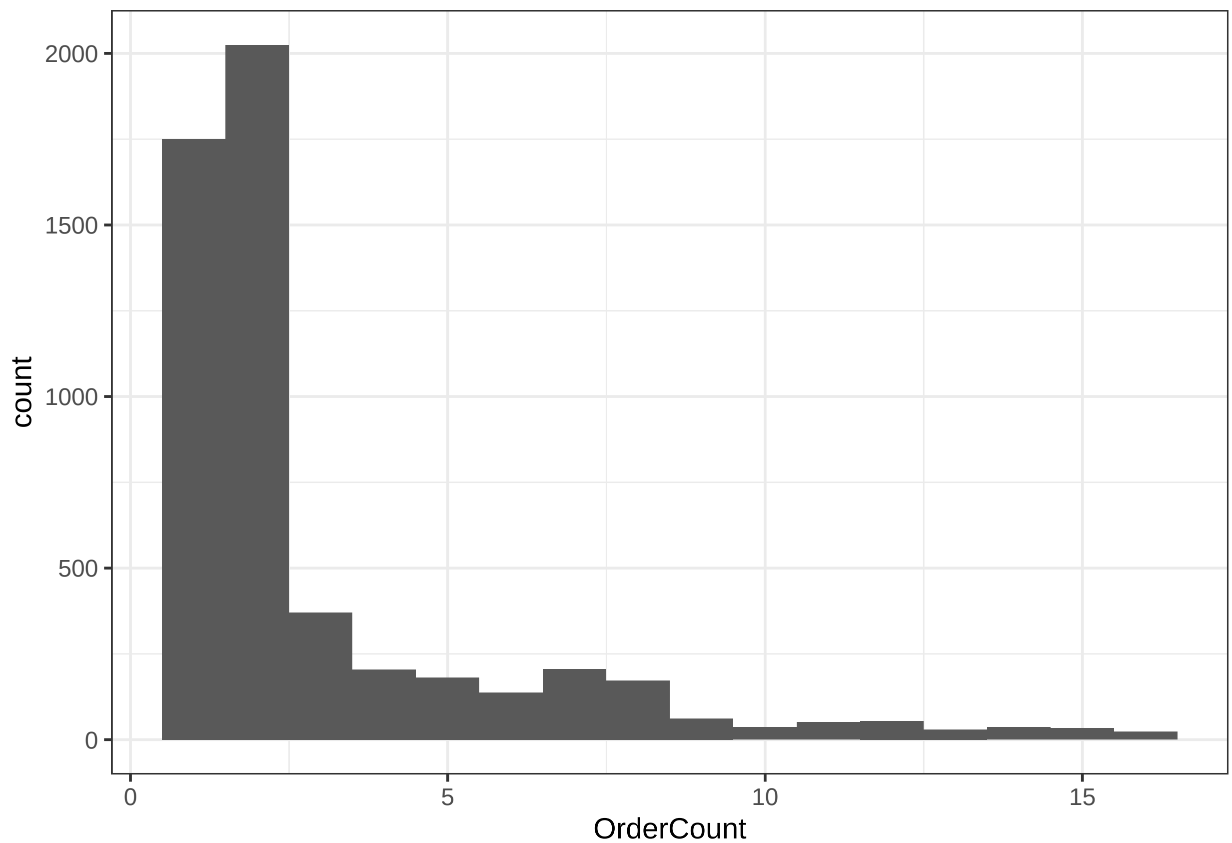

```
ggplot(dd, aes(CouponUsed)) + geom_histogram(binwidth = 1)
```

```
## Warning: Removed 256 rows containing non-finite values (stat_bin).
```



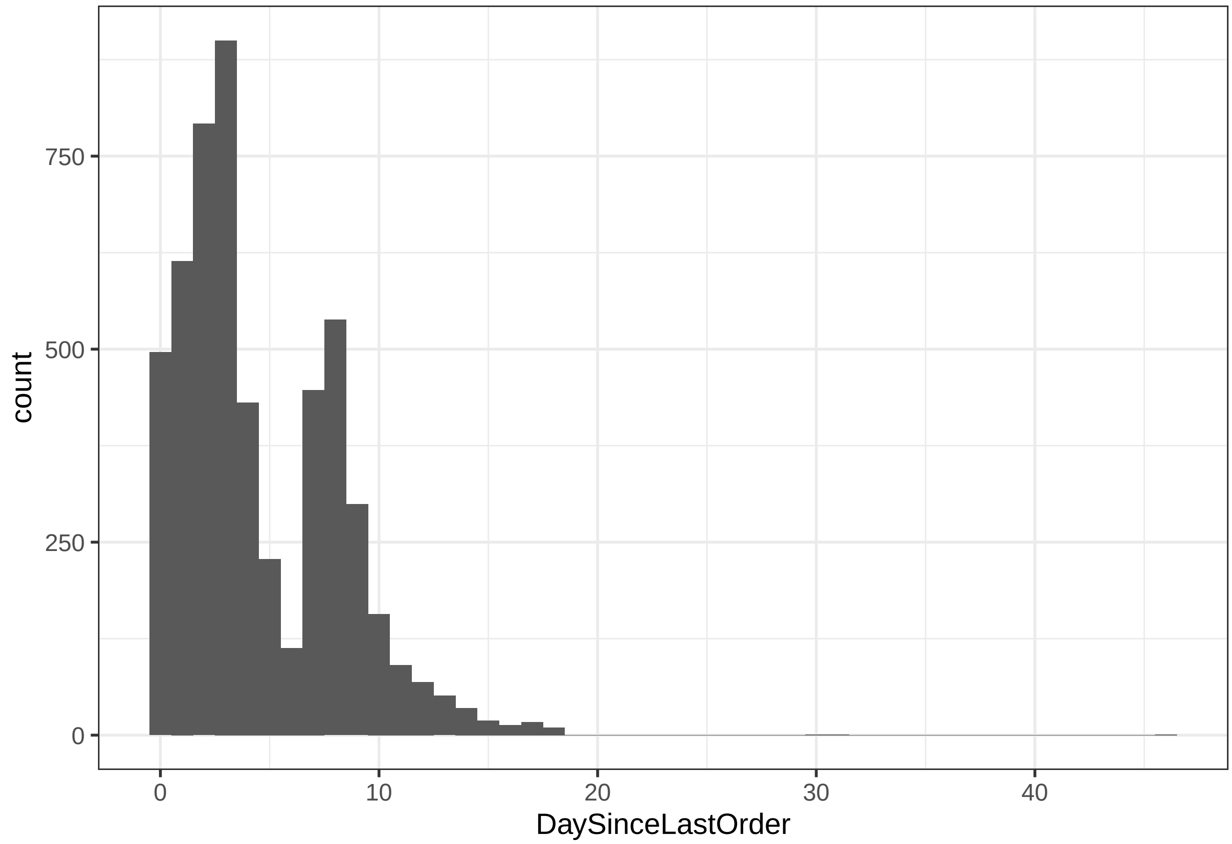

```
ggplot(dd, aes(OrderCount)) + geom_histogram(binwidth = 1)
```

```
## Warning: Removed 258 rows containing non-finite values (stat_bin).
```




```
ggplot(dd, aes(DaySinceLastOrder)) + geom_histogram(binwidth = 1)
```

```
## Warning: Removed 307 rows containing non-finite values (stat_bin).
```



Fill null values: when data distribution is closed to normal distribution, use mean to fill in; when data comprised of outliers, use median to fill in; when the data has more occurrences of a particular value, use mode to fill in.

```
dd$HourSpendOnApp[is.na(dd$HourSpendOnApp)]<-mean(dd$HourSpendOnApp,na.rm=TRUE)
dd[,'HourSpendOnApp']=round(dd[,'HourSpendOnApp'],0)
dd$DaySinceLastOrder[is.na(dd$DaySinceLastOrder)]<-median(dd$DaySinceLastOrder,na.rm=TRUE)
dd$OrderAmountHikeFromlastYear[is.na(dd$OrderAmountHikeFromlastYear)]<-median(dd$OrderAmountHikeFromlas
dd$WarehouseToHome[is.na(dd$WarehouseToHome)]<-median(dd$WarehouseToHome,na.rm=TRUE)
dd$OrderCount[is.na(dd$OrderCount)]<-median(dd$OrderCount,na.rm=TRUE)
dd$CouponUsed[is.na(dd$CouponUsed)]<-median(dd$CouponUsed,na.rm=TRUE)
dd$Tenure[is.na(dd$Tenure)] <- names(which.max(table(dd$Tenure,useNA="no")))
dd$Tenure <- as.numeric(dd$Tenure)

#check null values again
colSums(is.na(dd))
```

```
##                     Churn                    Tenure
##                         0                         0
##     PreferredLoginDevice                  CityTier
##                         0                         0
##          WarehouseToHome      PreferredPaymentMode
##                         0                         0
##                   Gender            HourSpendOnApp
##                         0                         0
##  NumberOfDeviceRegistered         PreferedOrderCat
```

```
##                            0                            0
##            SatisfactionScore                MaritalStatus
##                            0                            0
##              NumberOfAddress                     Complain
##                            0                            0
## OrderAmountHikeFromlastYear                   CouponUsed
##                            0                            0
##                   OrderCount            DaySinceLastOrder
##                            0                            0
##               CashbackAmount
##                            0
```

Get dummy

```
dd_dummy <- dummy_cols(dd, select_columns = c('PreferredLoginDevice','PreferredPaymentMode','Gender','P

# Delete the original categorical columns

dd_dummy[,c('PreferredLoginDevice','PreferredPaymentMode','Gender','PreferedOrderCat','MaritalStatus'):
```

Split data: test 20%, train 80%

```
set.seed(810)
test_index <- sample(nrow(dd_dummy), 1000)

test <- dd_dummy[test_index,]
train <- dd_dummy[-test_index,]


#split target column

test_x <- test[,-'Churn']
test_y <- test[,'Churn']
train_x <- train[,-'Churn']
train_y <- train[,'Churn']

# specific cleaning for regression

x_data <- model.matrix(Churn ~ -1 + ., dd_dummy)
y_data <- dd_dummy$Churn

x_train <- x_data[-test_index, ]
y_train <- y_data[-test_index]
x_test <- x_data[test_index, ]
y_test <- y_data[test_index]
```

Linear Regression

Fit a linear regression model with all features on 'Churn'

```r
f1 <- as.formula(Churn~.)
fit.lm1 <- lm(f1,train)
lm1.train.yhat <- predict(fit.lm1,train)
```

```
## Warning in predict.lm(fit.lm1, train): prediction from a rank-deficient fit may
## be misleading
```

```r
lm1.test.yhat <- predict(fit.lm1,test)
```

```
## Warning in predict.lm(fit.lm1, test): prediction from a rank-deficient fit may
## be misleading
```

Calculate MSE for the training dataset

```r
mse_train <- mean((train_y$Churn - lm1.train.yhat)^2)
mse_train
```

```
## [1] 0.09914056
```

Calculate MSE for the testing dataset

```r
mse_test <- mean((test_y$Churn - lm1.test.yhat)^2)
mse_test
```

```
## [1] 0.100005
```

Coefficient of f1

```r
coef(fit.lm1)
```

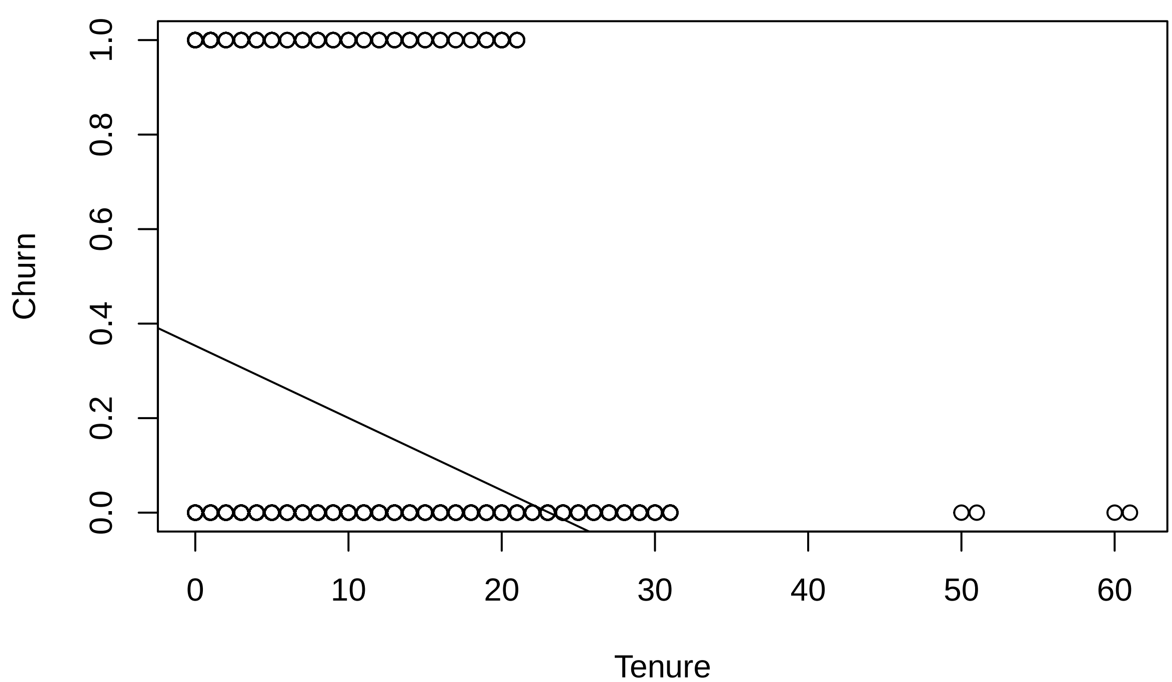
```
##                            (Intercept)                                 Tenure
##                           0.3532610101                          -0.0152976665
##                               CityTier                        WarehouseToHome
##                           0.0302061473                           0.0035720533
##                          HourSpendOnApp               NumberOfDeviceRegistered
##                          -0.0093099318                           0.0342326145
##                      SatisfactionScore                        NumberOfAddress
##                           0.0255757711                           0.0225309114
##                               Complain            OrderAmountHikeFromlastYear
##                           0.2095250555                           0.0005029148
##                             CouponUsed                             OrderCount
##                           0.0039089739                           0.0085809090
##                      DaySinceLastOrder                         CashbackAmount
##                          -0.0080140112                          -0.0011351549
##          PreferredLoginDevice_Computer      `PreferredLoginDevice_Mobile Phone`
##                           0.0334383248                          -0.0049618109
##             PreferredLoginDevice_Phone `PreferredPaymentMode_Cash on Delivery`
##                                     NA                           0.0497112078
##                PreferredPaymentMode_CC                PreferredPaymentMode_COD
##                          -0.0502441521                           0.1095816226
```

```
##      `PreferredPaymentMode_Credit Card`       `PreferredPaymentMode_Debit Card`
##                           -0.0047645173                            0.0150484955
##          `PreferredPaymentMode_E wallet`                PreferredPaymentMode_UPI
##                            0.0743936537                                      NA
##                           Gender_Female                              Gender_Male
##                           -0.0241086101                                      NA
##                PreferedOrderCat_Fashion                PreferedOrderCat_Grocery
##                           -0.1752825040                           -0.0737960962
##   `PreferedOrderCat_Laptop & Accessory`                 PreferedOrderCat_Mobile
##                           -0.3064028872                           -0.1960538095
##          `PreferedOrderCat_Mobile Phone`                 PreferedOrderCat_Others
##                           -0.1966559341                                      NA
##                  MaritalStatus_Divorced                   MaritalStatus_Married
##                           -0.1018610154                           -0.1160282870
##                    MaritalStatus_Single
##                                      NA
```

Plot regression on Tenure as the X-variable

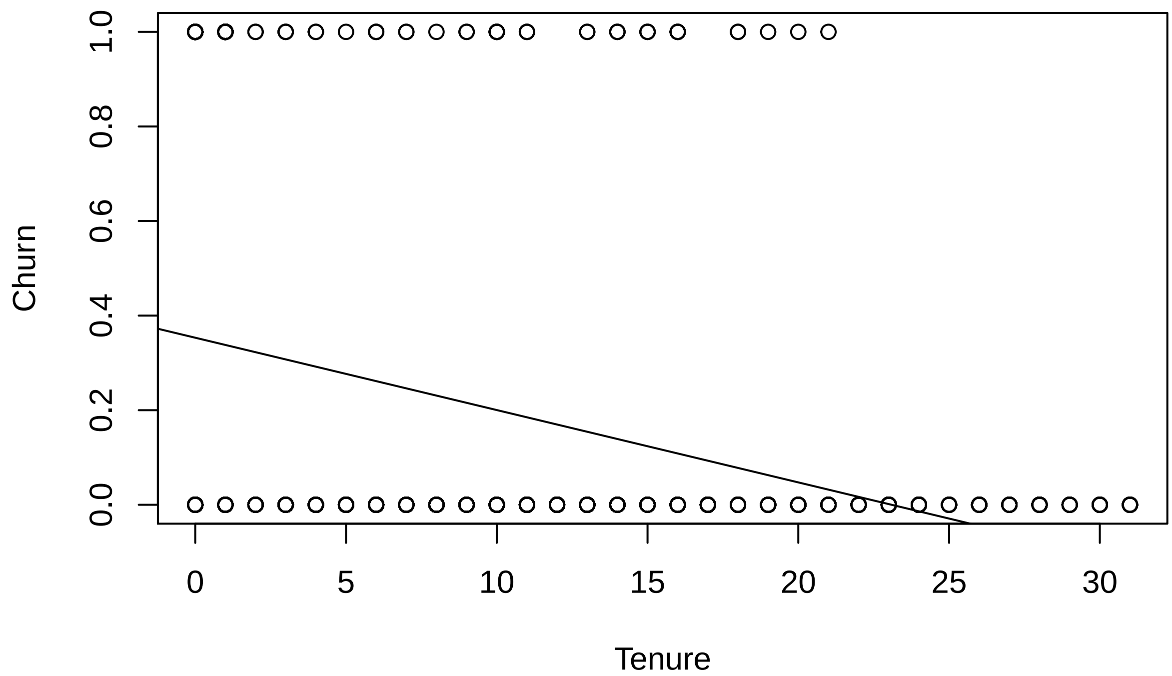
```r
plot(Churn~Tenure,data = train)
abline(fit.lm1)
```

```
## Warning in abline(fit.lm1): only using the first two of 35 regression
## coefficients
```



```r
plot(Churn~Tenure,data = test)
abline(fit.lm1)
```

```
## Warning in abline(fit.lm1): only using the first two of 35 regression
## coefficients
```



Yhat for either training or testing dataset can be within the range of neigative infinity to positive infinity. Yhat above 1, we will conclude that this customer will churn. Yhat between 0 - 1, we will conclude that yhat indicates the probability the customer will churn. Yhat below 0, we will conclude that this customer will not churn. The MSE for training dataset is 0.0991 and the MSE for testing dataset is 0.0999. Both MSEs are small, which means our linear regression model predicts the outcome well. However, the reason why MSEs are low is likely due to the fact that f1 contains too many features. That can also be a downside of our model, because it captures too much noise of the dataset. Thus, we want to reduce some dimensionalities but also capture the full information of our dataset. There is a trade off between variances and biases when we remove features. Next in our project, we'll use Ridge/Lasso to help us maintain the balance between variances and biases. The ultimate goal is to remove insignificant variables, and still minimize the MSE.

Ridge Regression

Use cross validation to select the best lambda on the training data:

```r
fit.ridge <- cv.glmnet(x_train, y_train, alpha = 0, nfolds = 10)
```

Checked the iteration of lambda:

```
lam <- fit.ridge$lambda
min(lam)
```

```
## [1] 0.01304079
```

The least lambda = 0.013.

Calculate the MSE for training data:

```
yhat.train.ridge <- predict(fit.ridge, x_train, s = fit.ridge$lambda.min)
mse.train.ridge <- mean((y_train - yhat.train.ridge)^2)
mse.train.ridge
```

```
## [1] 0.09924953
```

Training MSE = 0.0992.

```
yhat.test.ridge <- predict(fit.ridge, x_test, s = fit.ridge$lambda.min)
mse.test.ridge <- mean((y_test - yhat.test.ridge)^2)
mse.test.ridge
```

```
## [1] 0.09987271
```

Testing MSE = 0.0998, just slightly higher than the training MSE.

Observe the coefficient output:

```
coef(fit.ridge, s=min(lam))
```

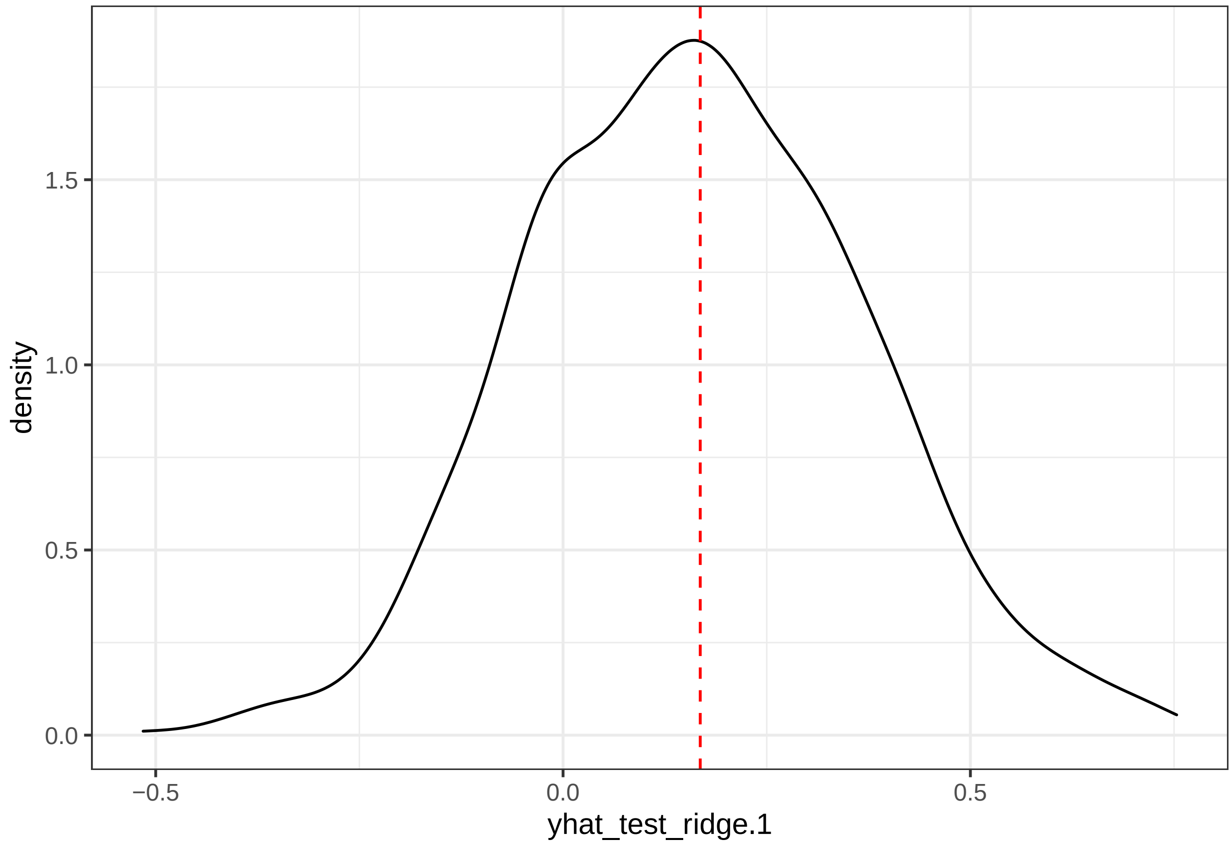
```
## 35 x 1 sparse Matrix of class "dgCMatrix"
##                                                 1
## (Intercept)                          0.0408013700
## Tenure                              -0.0144894206
## CityTier                             0.0298180710
## WarehouseToHome                      0.0034652164
## HourSpendOnApp                      -0.0114077587
## NumberOfDeviceRegistered             0.0319677907
## SatisfactionScore                    0.0248925524
## NumberOfAddress                      0.0209438169
## Complain                             0.2020952616
## OrderAmountHikeFromlastYear          0.0003899849
## CouponUsed                           0.0040415230
## OrderCount                           0.0079688038
## DaySinceLastOrder                   -0.0078144800
## CashbackAmount                      -0.0007030125
## PreferredLoginDevice_Computer        0.0240598186
## `PreferredLoginDevice_Mobile Phone` -0.0157477925
## PreferredLoginDevice_Phone          -0.0057599392
## `PreferredPaymentMode_Cash on Delivery` 0.0265096045
## PreferredPaymentMode_CC             -0.0660482204
## PreferredPaymentMode_COD             0.0849311443
```

```
## `PreferredPaymentMode_Credit Card`      -0.0256499686
## `PreferredPaymentMode_Debit Card`       -0.0065139570
## `PreferredPaymentMode_E wallet`          0.0518861629
## PreferredPaymentMode_UPI                -0.0198025057
## Gender_Female                           -0.0117636213
## Gender_Male                              0.0116890132
## PreferedOrderCat_Fashion                 0.0169646314
## PreferedOrderCat_Grocery                 0.0850465043
## `PreferedOrderCat_Laptop & Accessory`   -0.0926806822
## PreferedOrderCat_Mobile                  0.0282174764
## `PreferedOrderCat_Mobile Phone`          0.0251295528
## PreferedOrderCat_Others                  0.1409935559
## MaritalStatus_Divorced                  -0.0308760473
## MaritalStatus_Married                   -0.0453662431
## MaritalStatus_Single                     0.0695384736
```

Everything else equal, for each extra year that the customer stayed with the company, as signaled by the Tenure variable, the probability of churn is 1.45% lower, which is quite intuitive. And if a customer has complained, the probability of churn is 20% higher.

Plot the test distribution of churn prediction:

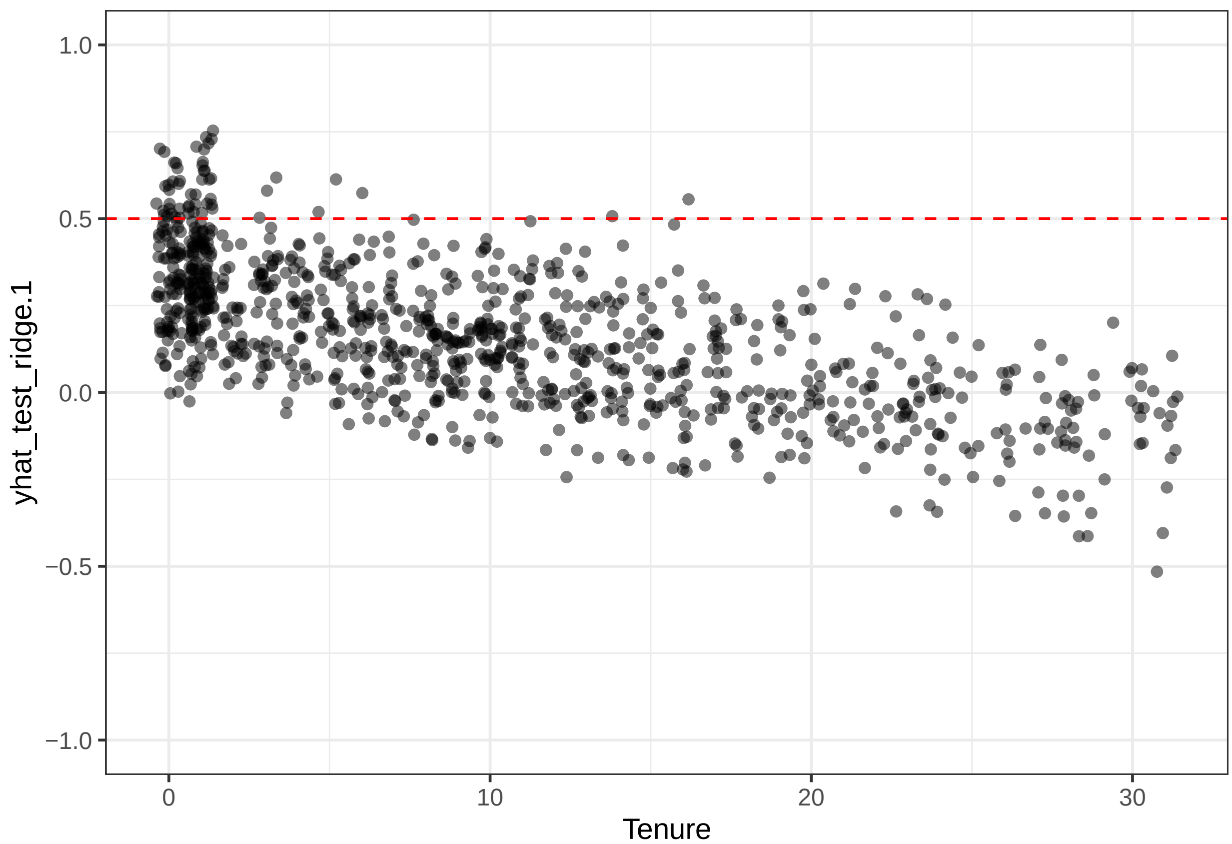
```
x_test_df = as.data.frame(x_test)

pred_ridge <- data.table(yhat_test_ridge = yhat.test.ridge, Complain = x_test_df$Complain, Tenure = x_te

ggplot(pred_ridge, aes(yhat_test_ridge.1)) + geom_density() +
  geom_vline(xintercept = mean(dd$Churn), color = "red", linetype = 2)
```



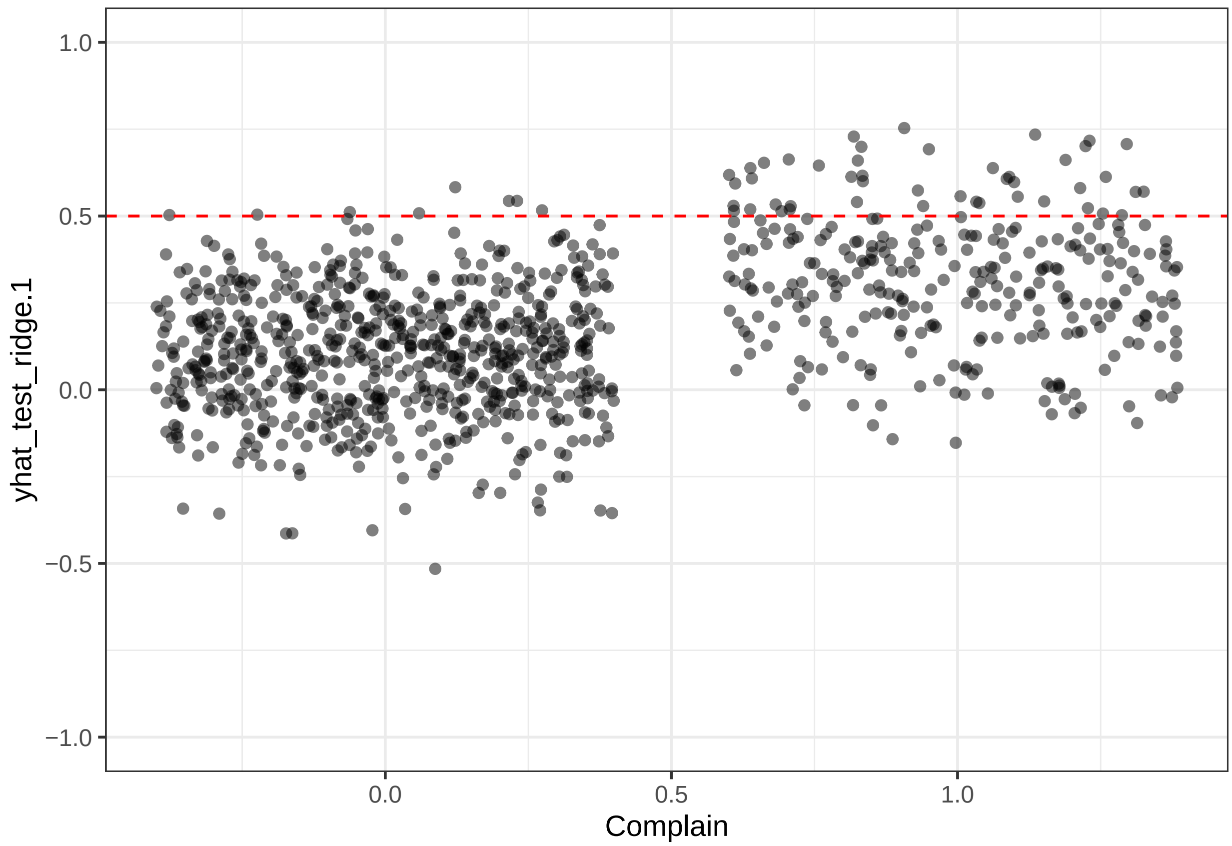
Since we have around 17% churn rate, the ridge regression also has a center and peak at about 17%. The majority of the predictions almost symmetrically fall in the range from 0 to 0.375. In practice, we will set threshold = 0.5 to determine whether a customer will churn or not. Arbitrarily for any customer that has probability lower than 0.5, we predict no churn; and higher than 0.5, we predict churn.

Plot the churn prediction against some features:

```
ggplot(pred_ridge, aes(Tenure, yhat_test_ridge.1, 0)) + geom_jitter(alpha=0.5) + ylim(-1,1) +
  geom_hline(yintercept = 0.5, color = "red", linetype = 2)
```




```
ggplot(pred_ridge, aes(Complain, yhat_test_ridge.1, 0)) + geom_jitter(alpha=0.5) + ylim(-1,1) +
  geom_hline(yintercept = 0.5, color = "red", linetype = 2)
```



Most of the observations have a tenure of 0-2 years, while for the rest there is a negative correlation between tenure and probability of churn. Similarly, most customers did not have complaint records, thus very few observations have predicted probabilities that are above 0.5, while for those that have complained, the likelihood is higher.

Lasso Regression

Fitting model is similar to ridge regression:

```
fit.lasso <- cv.glmnet(x_train, y_train, alpha = 1, nfolds=10)
```

Training MSE = 0.0992.

Checked the iteration of lambda:

```
lam_1 <- fit.lasso$lambda
min(lam_1)
```

```
## [1] 0.0001464904
```

The least lambda = 0.00016.

Calculate the MSE for training data:

```r
yhat.train.lasso <- predict(fit.lasso, x_train, s = fit.lasso$lambda.min)
mse.train.lasso <- mean((y_train - yhat.train.lasso)^2)
mse.train.lasso
```

```
## [1] 0.09919132
```

Training MSE = 0.0992.

```r
yhat.test.lasso <- predict(fit.lasso, x_test, s = fit.lasso$lambda.min)
mse.test.lasso <- mean((y_test - yhat.test.lasso)^2)
mse.test.lasso
```

```
## [1] 0.09995879
```

Testing MSE = 0.0999, also higher than the training MSE.

Inspect the coefficients:

```r
coef(fit.lasso, fit.lasso$lambda.min)
```

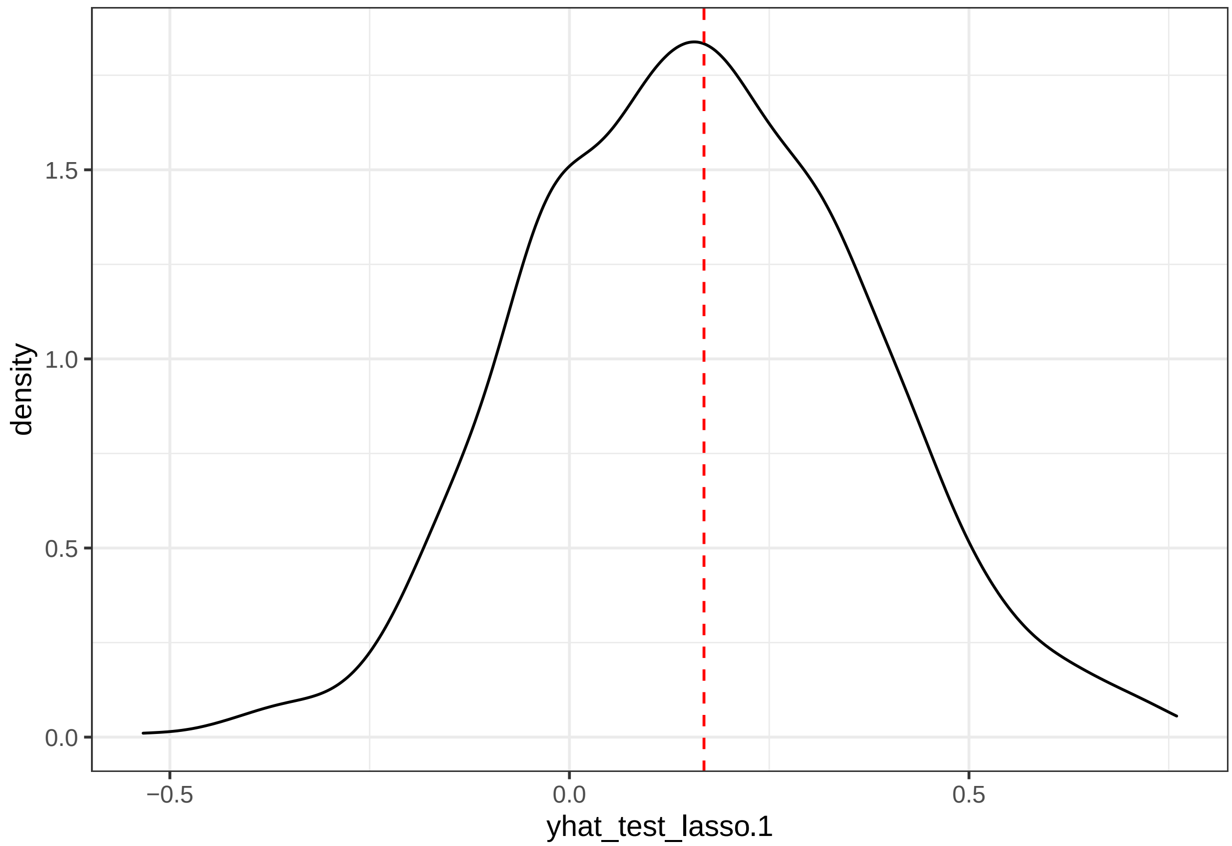
```
## 35 x 1 sparse Matrix of class "dgCMatrix"
##                                                  1
## (Intercept)                            4.320604e-02
## Tenure                                -1.509697e-02
## CityTier                               2.985282e-02
## WarehouseToHome                        3.470531e-03
## HourSpendOnApp                        -9.925310e-03
## NumberOfDeviceRegistered               3.216858e-02
## SatisfactionScore                      2.496424e-02
## NumberOfAddress                        2.179147e-02
## Complain                               2.071746e-01
## OrderAmountHikeFromlastYear            1.715390e-04
## CouponUsed                             3.418151e-03
## OrderCount                             8.245681e-03
## DaySinceLastOrder                     -7.679368e-03
## CashbackAmount                        -7.971702e-04
## PreferredLoginDevice_Computer          2.999839e-02
## `PreferredLoginDevice_Mobile Phone`   -7.603427e-03
## PreferredLoginDevice_Phone             .
## `PreferredPaymentMode_Cash on Delivery` 2.912393e-02
## PreferredPaymentMode_CC               -5.713366e-02
## PreferredPaymentMode_COD               9.179582e-02
## `PreferredPaymentMode_Credit Card`    -1.820409e-02
## `PreferredPaymentMode_Debit Card`      .
## `PreferredPaymentMode_E wallet`        5.748126e-02
## PreferredPaymentMode_UPI              -1.094067e-02
## Gender_Female                         -2.189737e-02
## Gender_Male                            3.557349e-13
## PreferedOrderCat_Fashion               .
## PreferedOrderCat_Grocery               7.659981e-02
## `PreferedOrderCat_Laptop & Accessory` -1.167100e-01
## PreferedOrderCat_Mobile                .
```

```
## `PreferedOrderCat_Mobile Phone`           .
## PreferedOrderCat_Others                   1.364180e-01
## MaritalStatus_Divorced                    .
## MaritalStatus_Married                    -1.320365e-02
## MaritalStatus_Single                      1.012387e-01
```

It shrank coefficients for some variables to zero, such as debit card payment, gender male, and preferred order categories. As a result, the absolute value of most other coefficients slightly increased, like we see here for Tenure and Complain. The training and testing MSE are roughly the same as ridge regression.

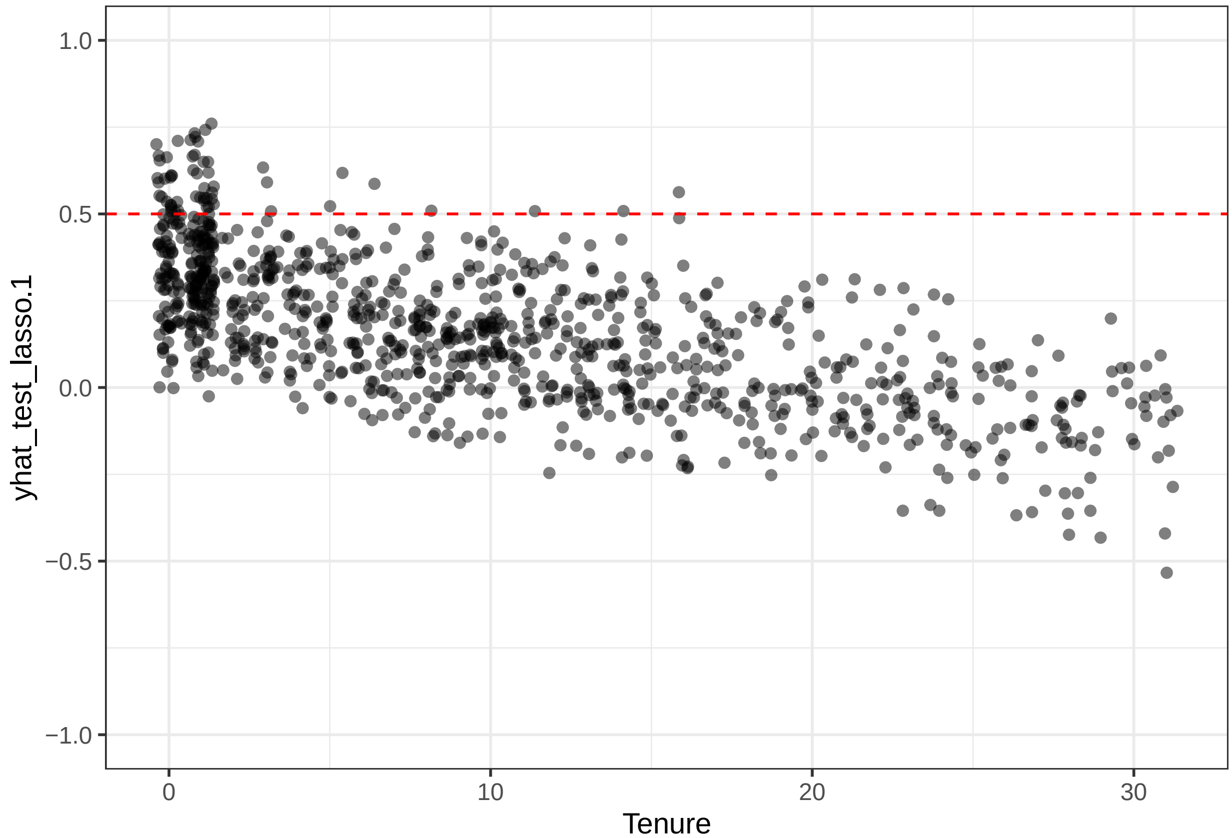
Plot the test predictions

```r
pred_lasso <- data.table(yhat_test_lasso = yhat.test.lasso, Complain = x_test_df$Complain, Tenure = x_te

ggplot(pred_lasso, aes(yhat_test_lasso.1)) + geom_density() +
  geom_vline(xintercept = mean(dd$Churn), color = "red", linetype = 2)
```

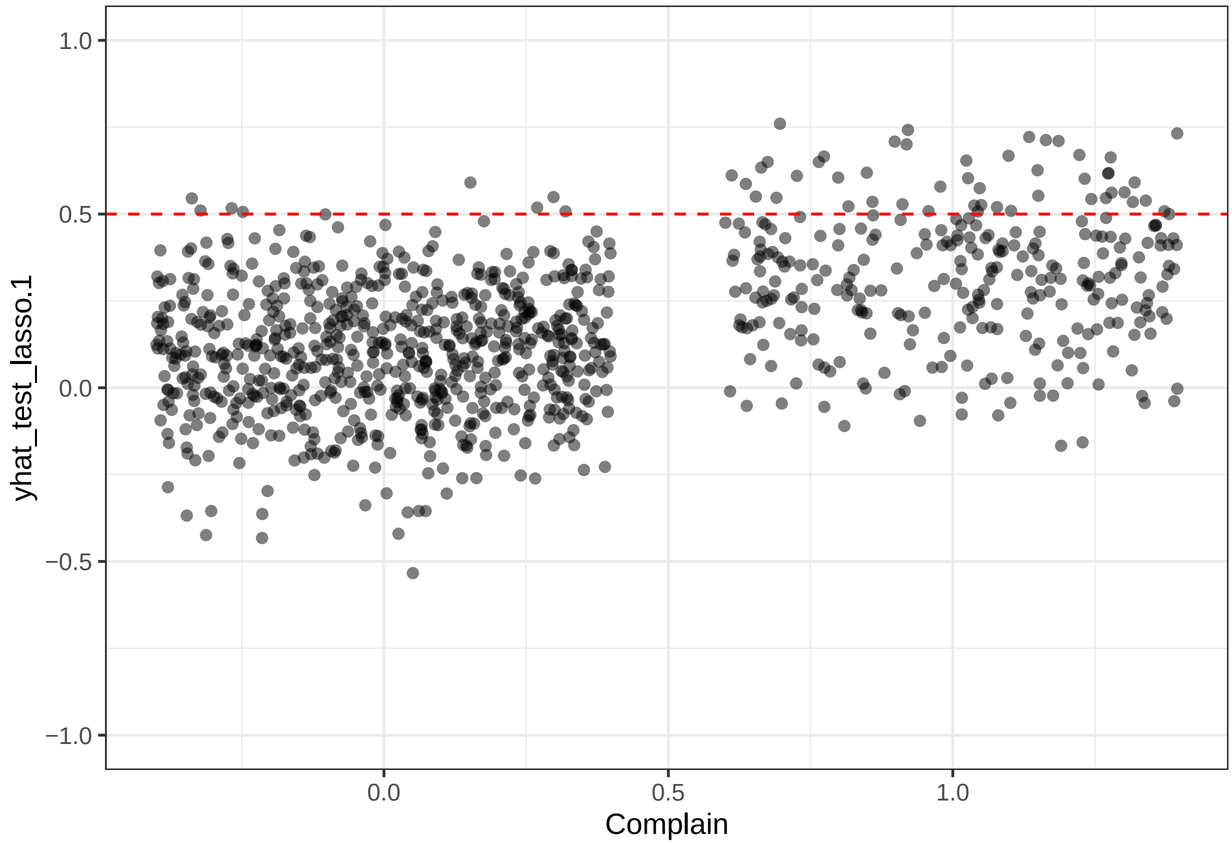


Plot the churn prediction against some features:

```r
ggplot(pred_lasso, aes(Tenure, yhat_test_lasso.1, 0)) + geom_jitter(alpha=0.5) + ylim(-1,1) +
  geom_hline(yintercept = 0.5, color = "red", linetype = 2)
```




```
ggplot(pred_lasso, aes(Complain, yhat_test_lasso.1, 0)) + geom_jitter(alpha=0.5) + ylim(-1,1) +
  geom_hline(yintercept = 0.5, color = "red", linetype = 2)
```



The results and explanation are very similar to those for the ridge regression, meaning that these two models do not have significantly different performance on our data.

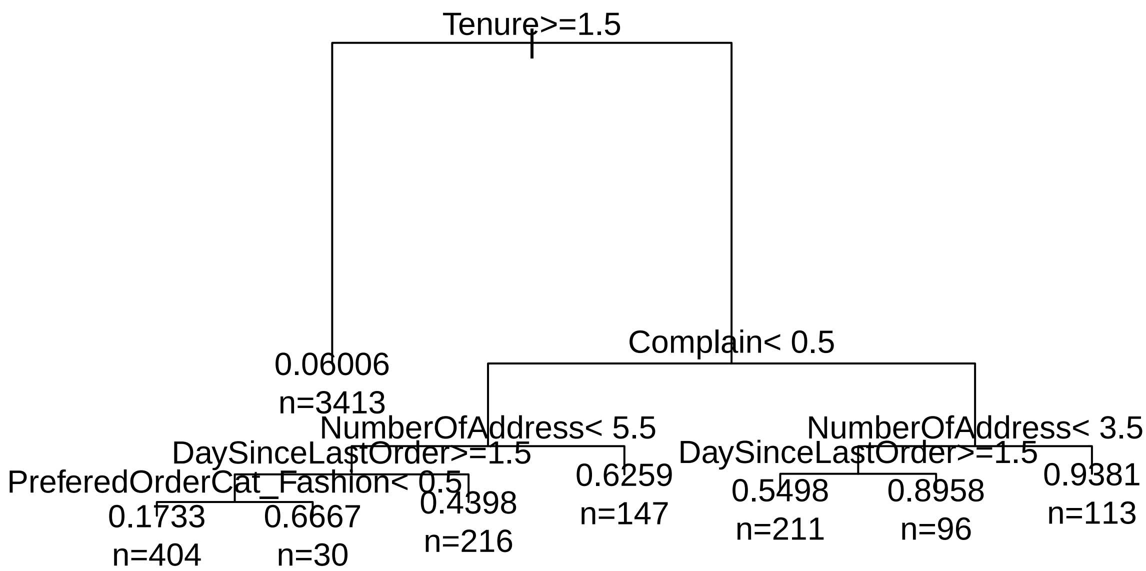
Regression Tree

Build a regression tree with cp=0.01

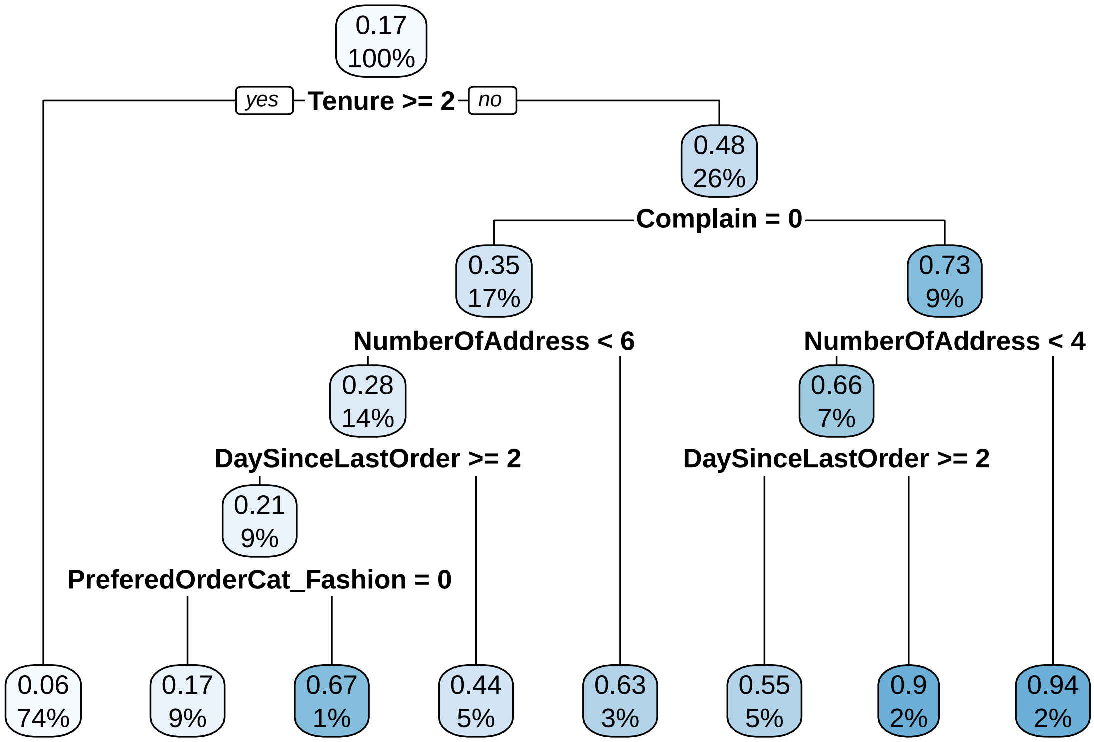
```
f1 <- as.formula(Churn ~ .)
fit.tree <- rpart(f1,
                  train,
                  control = rpart.control(cp = 0.01))
```

plot this tree

```
par(xpd = TRUE)
plot(fit.tree, compress=TRUE)
text(fit.tree, use.n=TRUE)
```




```
rpart.plot(fit.tree)
```

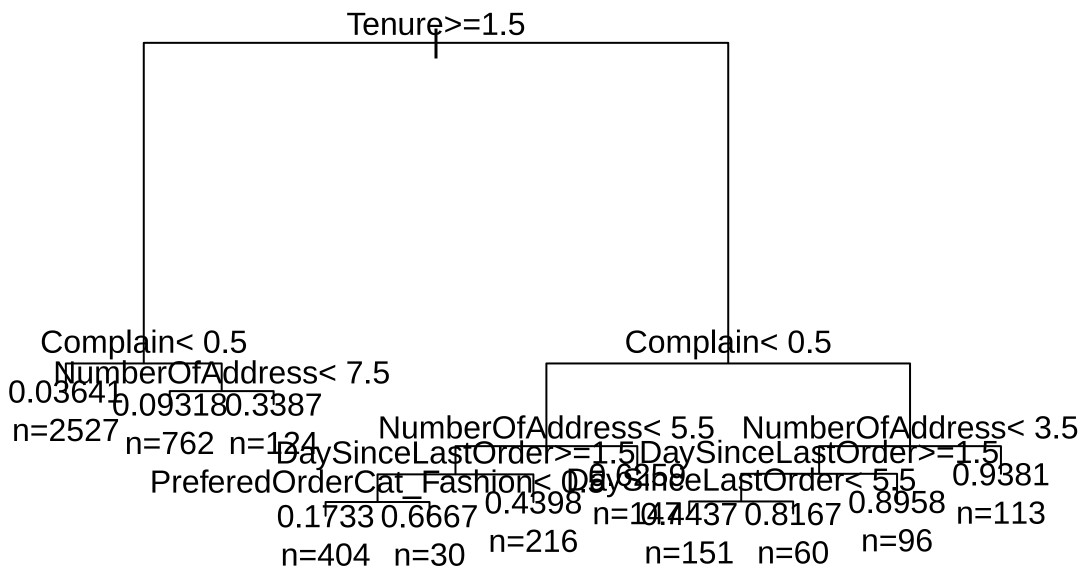


Build a regression tree with smaller cp

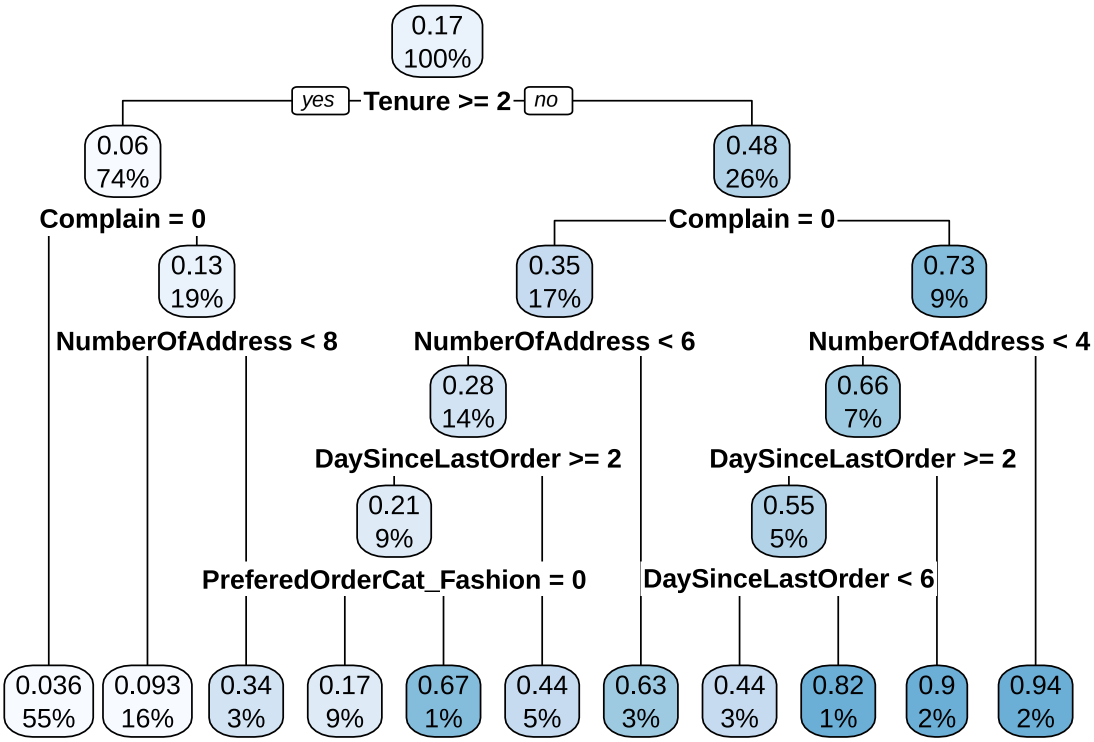
```
fit.tree2 <- rpart(f1,
                   train,
                   control = rpart.control(cp = 0.008))
```

plot this tree

```
par(xpd = TRUE)
plot(fit.tree2, compress=TRUE)
text(fit.tree2, use.n=TRUE)
```




```
rpart.plot(fit.tree2)
```



training set mse

```
yhat.tree <- predict(fit.tree, train)
mse.tree.train <- colMeans((yhat.tree - train_y) ^ 2)
print(mse.tree.train)
```

```
##      Churn
## 0.08911799
```

testing set mse

```
yhat.tree <- predict(fit.tree, test)
mse.tree.test <- colMeans((yhat.tree - test_y) ^ 2)
print(mse.tree.test)
```

```
##      Churn
## 0.08854205
```

Conclusion: The first rule is related to Tenure. If the value of tenure is greater and equal to 2, the chance of churn is 6%. It makes sense. A longer tenure more than 2 years means that customers are likely to stay and have a great user experience. We can assume that tenure is one of the most essential factors we can use to determine whether customers might churn. Let's dive into right side. If customer has a shorter tenure, some factors such as complain, number of address and the day since last order play a crucial rule. For example, customers who have no complain but more addresses and last orders are less likely to churn. Once again, it indicates that user experience bring a significant effect to customers. In addition, even if not loyal customers tend to make orders no matter where they were.

Bagging

Data Cleaning

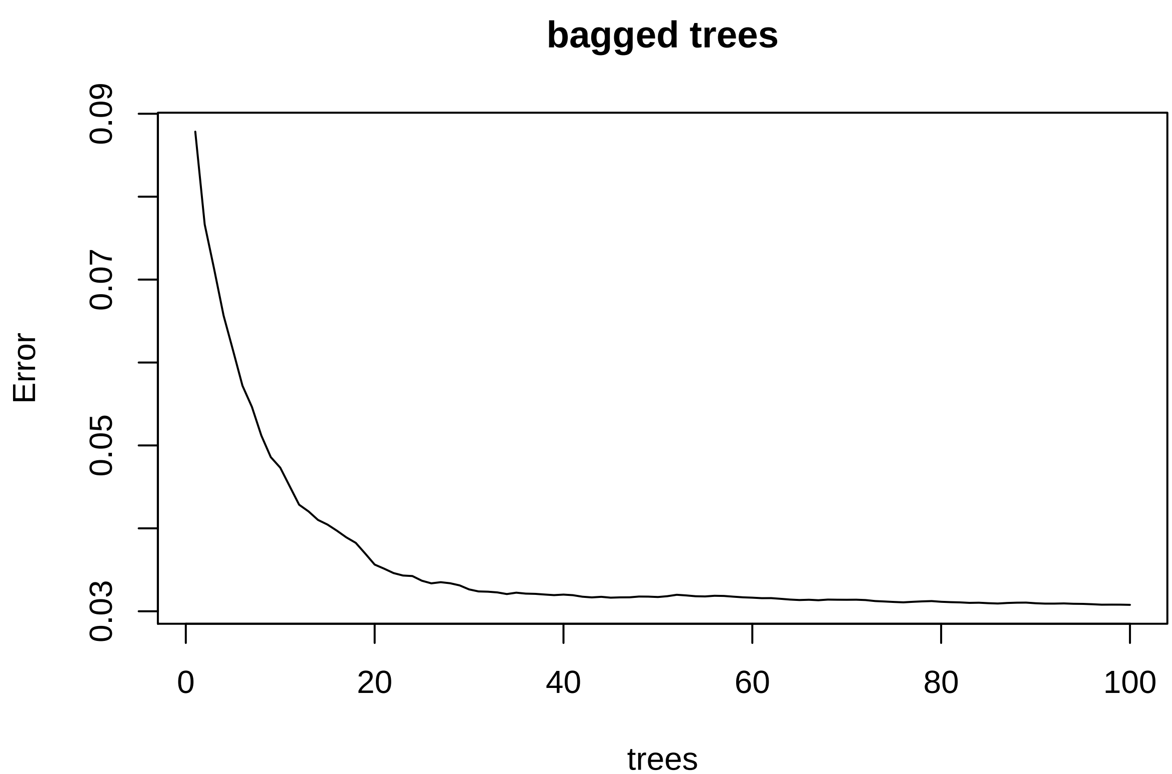
```
dd_dummy2<-dd_dummy[,]
#dd_dummy2$Churn<-as.factor(dd_dummy2$Churn)
names(dd_dummy2)[16]<-"PreferredLoginDevice_Mobile_Phone"
names(dd_dummy2)[18]<-"PreferredPaymentMode_Cash_on_Delivery"
names(dd_dummy2)[21]<-"PreferredPaymentMode_Credit_Card"
names(dd_dummy2)[22]<-"PreferredPaymentMode_Debit_Card"
names(dd_dummy2)[23]<-"PreferredPaymentMode_E_wallet"
names(dd_dummy2)[29]<-"PreferedOrderCat_Laptop_Accessory"
names(dd_dummy2)[31]<-"PreferedOrderCat_Mobile_Phone"
```

bagging model

```
dftest<-dd_dummy2[test_index,]#train
dftrain<-dd_dummy2[-test_index,]#test
model <- randomForest(Churn~.,dftrain,mtry=34,ntree=100)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
plot(model,main="bagged trees")
```

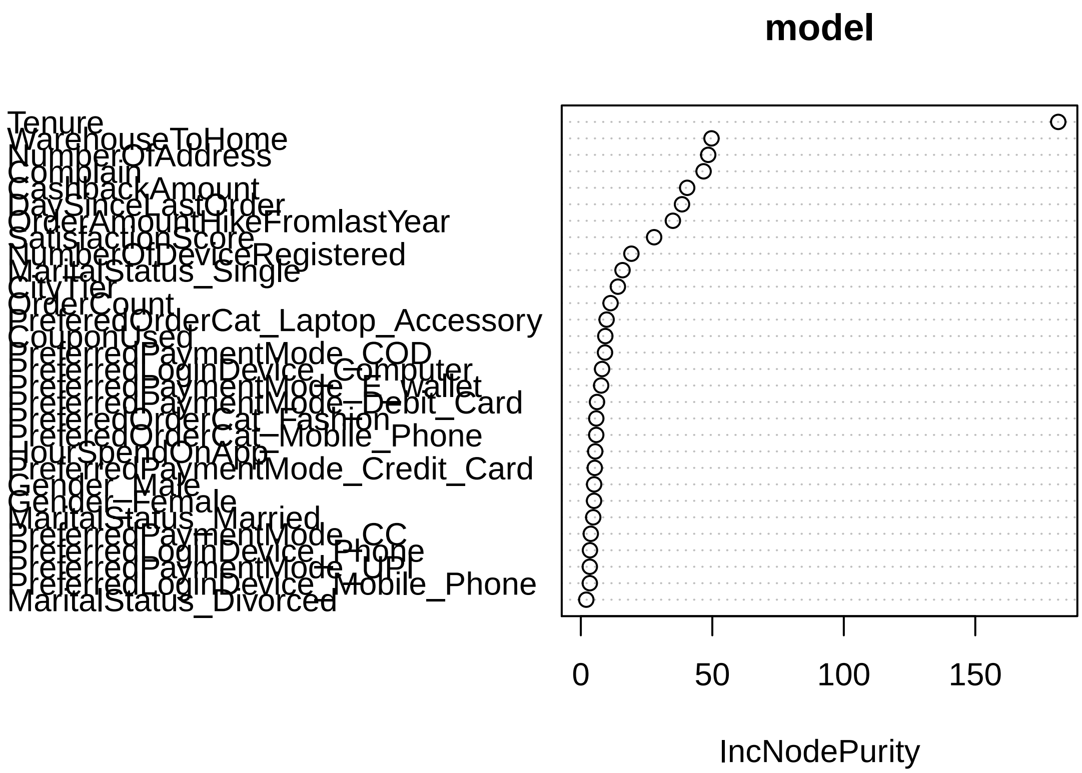


Variable importance for 200 bagged trees for the Ecommerce Data

```r
importance(model)
```

```
##                                          IncNodePurity
## Tenure                                      181.551762
## CityTier                                     14.086607
## WarehouseToHome                              49.698868
## HourSpendOnApp                                5.472617
## NumberOfDeviceRegistered                     19.233205
## SatisfactionScore                            27.869799
## NumberOfAddress                              48.435631
## Complain                                     46.685373
## OrderAmountHikeFromlastYear                  35.007450
## CouponUsed                                    9.306625
## OrderCount                                   11.304948
## DaySinceLastOrder                            38.473768
## CashbackAmount                               40.444563
## PreferredLoginDevice_Computer                 8.083306
## PreferredLoginDevice_Mobile_Phone             3.407596
## PreferredLoginDevice_Phone                    3.472797
## PreferredPaymentMode_Cash_on_Delivery         1.396512
## PreferredPaymentMode_CC                       3.778517
## PreferredPaymentMode_COD                      9.198213
## PreferredPaymentMode_Credit_Card              5.296285
## PreferredPaymentMode_Debit_Card               6.153810
## PreferredPaymentMode_E_wallet                 7.723491
## PreferredPaymentMode_UPI                      3.410912
## Gender_Female                                 5.029039
## Gender_Male                                   5.053557
## PreferedOrderCat_Fashion                      5.875979
## PreferedOrderCat_Grocery                      1.115174
## PreferedOrderCat_Laptop_Accessory             9.807030
## PreferedOrderCat_Mobile                       1.728436
## PreferedOrderCat_Mobile_Phone                 5.861404
## PreferedOrderCat_Others                       0.691520
## MaritalStatus_Divorced                        2.077393
## MaritalStatus_Married                         4.708620
## MaritalStatus_Single                         15.860541
```

```r
varImpPlot(model)
```



ROC Curve 0.5<roc<1, Better than random guessing. This classifier (model) can have predictive value if the threshold value is properly set

```
pre<-predict(model,dftest)
pre <- as.numeric(pre)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

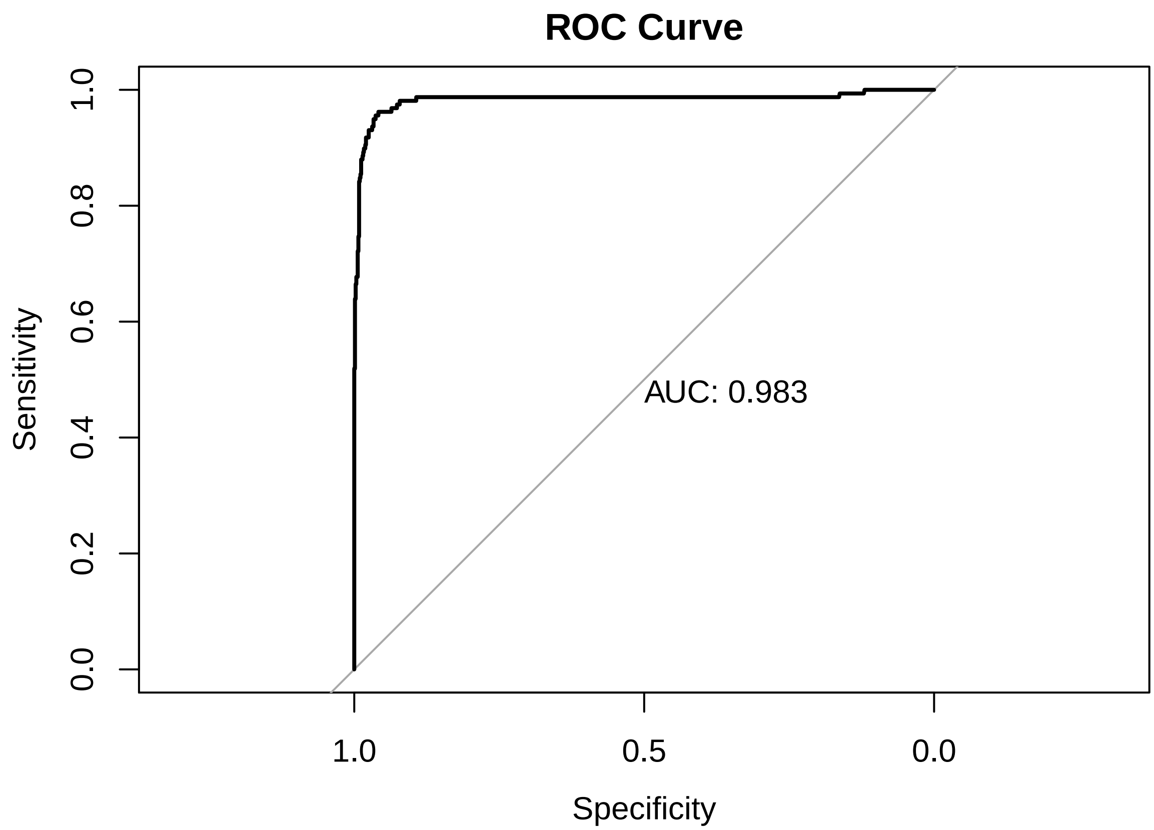
```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
rocbagging<-roc(dftest$Churn,pre,plot=T,print.auc=T,main="ROC Curve")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



Finding the best threshold value and confusion matrix

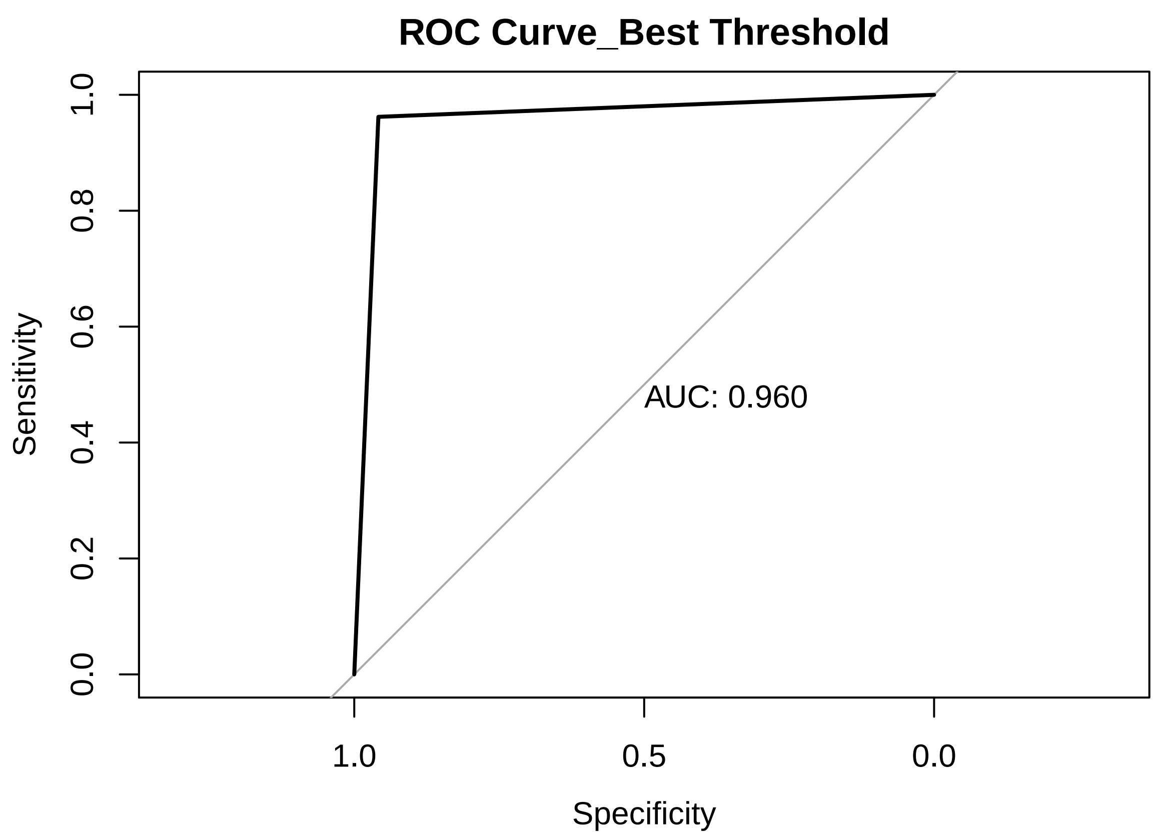
```
bestdd<-coords(rocbagging,"best")[1,1]
preclass<-ifelse(pre>bestdd,1,0)
ma<-table(true=dftest$Churn,preclass)
ma
```

```
##      preclass
## true   0   1
##    0 807  35
##    1   6 152
```

```
roc(dftest$Churn,preclass,plot=T,print.auc=T,main="ROC Curve_Best Threshold")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = dftest$Churn, predictor = preclass, plot = T,     print.auc = T, main = "ROC C
##
## Data: preclass in 842 controls (dftest$Churn 0) < 158 cases (dftest$Churn 1).
## Area under the curve: 0.9602
```

Accuracy for Bagging Model

```
sum(diag(ma))/sum(ma)
```

```
## [1] 0.959
```

MSE TRAIN

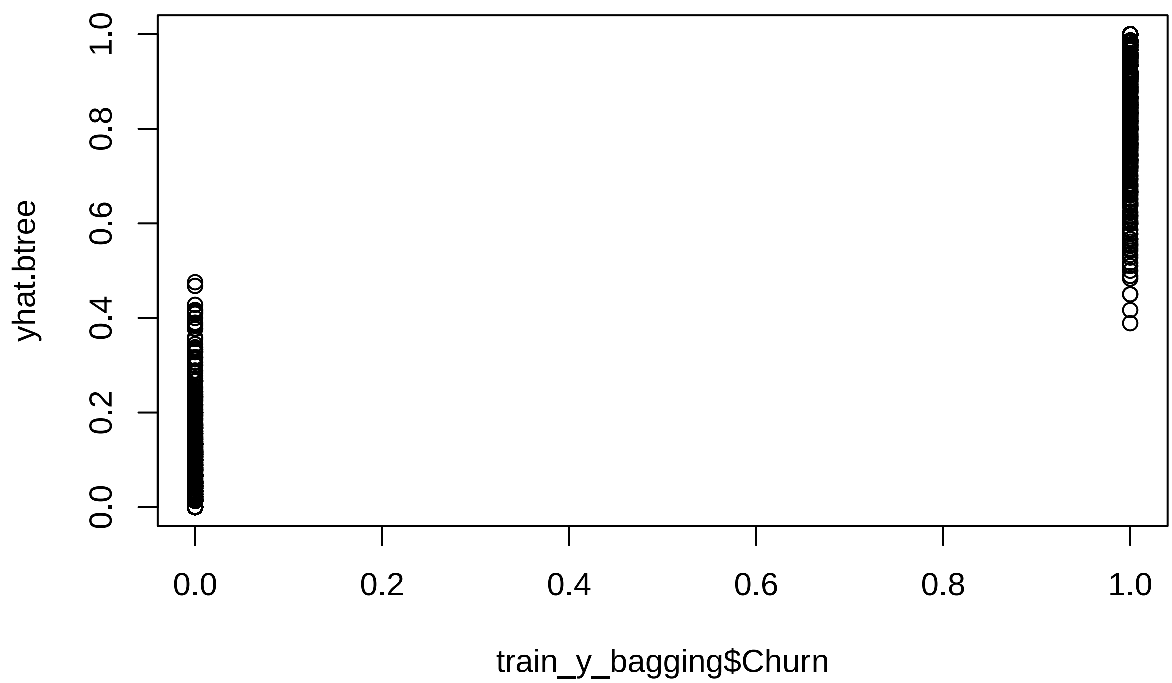
```
dd_dummy2$Churn<-as.integer(dd_dummy2$Churn)
dftest<-dd_dummy2[test_index,]
dftrain<-dd_dummy2[-test_index,]
model <- randomForest(Churn~.,dftrain,mtry=34,ntree=15)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
test_x_bagging <- dftest[,-1]
test_y_bagging <- dftest[,'Churn']
train_x_bagging <- dftrain[,-1]
train_y_bagging <- dftrain[,'Churn']
yhat.btree <- predict(model, dftrain, n.trees = 15)
mse.btree <- colMeans((yhat.btree - train_y_bagging) ^ 2)
print(mse.btree)
```

```
##      Churn
## 0.00786485
```

```
plot(train_y_bagging$Churn, yhat.btree)
```

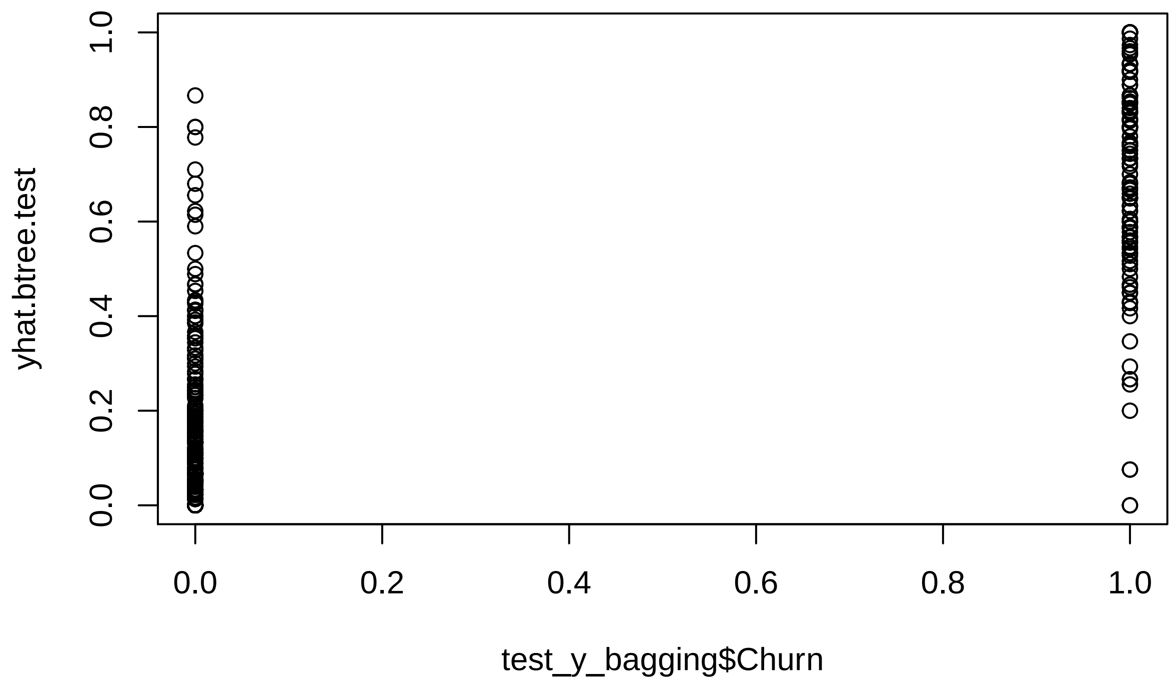


MSE Test

```
yhat.btree.test <- predict(model, dftest, n.trees = 15)
mse.btree.test <- colMeans((yhat.btree.test - test_y_bagging) ^ 2)
print(mse.btree.test)
```

```
##      Churn
## 0.03277165
```

```
plot(test_y_bagging$Churn, yhat.btree.test)
```



Conclusion: As we add more trees we're averaging over more high variance decision trees. Early on, we see a dramatic reduction in variance (and hence our error) but eventually the error flatting and stabilize signaling at 50 trees. Also. we'll likely not gain much improvement by bagging more trees after 50 trees. Bagging Model has a really good performance no matter in AUC Curve and MSE in Training and Testing. When comparing with the other five models, bagging model still has significantly lower MSE results than others. We guess it might be because this model allow each round to use bootstraping's method to extract 34 training samples except Churn from the original sample set.

Boosting

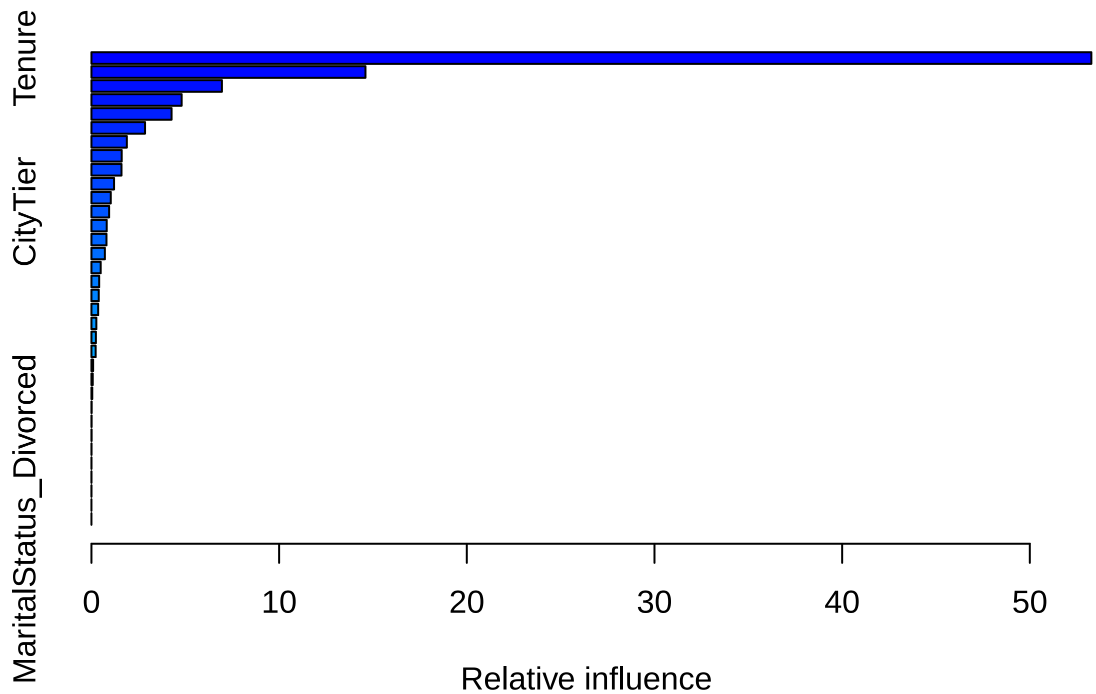
```
set.seed(810)
f1 <- as.formula(Churn ~ .)
```

Because we used a comparatively small lambda, we used more trees to converge to the desired result.

```
fit.boost_tree <- gbm(f1,
                  data = train,
                  distribution = "gaussian",
                  n.trees = 10000,
                  interaction.depth = 2,
                  shrinkage = 0.001)
```

Show relative influence

```
summary(fit.boost_tree)
```




```
##                                                                        var
## Tenure                                                               Tenure
## Complain                                                           Complain
## NumberOfAddress                                             NumberOfAddress
## DaySinceLastOrder                                         DaySinceLastOrder
## MaritalStatus_Single                                   MaritalStatus_Single
## NumberOfDeviceRegistered                           NumberOfDeviceRegistered
## WarehouseToHome                                             WarehouseToHome
## CashbackAmount                                               CashbackAmount
## SatisfactionScore                                         SatisfactionScore
## `PreferedOrderCat_Laptop & Accessory`   `PreferedOrderCat_Laptop & Accessory`
## OrderCount                                                       OrderCount
## CityTier                                                           CityTier
## `PreferredPaymentMode_E wallet`               `PreferredPaymentMode_E wallet`
## CouponUsed                                                       CouponUsed
## PreferredPaymentMode_COD                           PreferredPaymentMode_COD
## OrderAmountHikeFromlastYear                     OrderAmountHikeFromlastYear
## PreferedOrderCat_Fashion                           PreferedOrderCat_Fashion
## PreferredLoginDevice_Computer                 PreferredLoginDevice_Computer
## `PreferedOrderCat_Mobile Phone`               `PreferedOrderCat_Mobile Phone`
## PreferredPaymentMode_CC                             PreferredPaymentMode_CC
```

```
## `PreferredPaymentMode_Credit Card`          `PreferredPaymentMode_Credit Card`
## MaritalStatus_Married                                    MaritalStatus_Married
## PreferredLoginDevice_Phone                          PreferredLoginDevice_Phone
## Gender_Female                                                    Gender_Female
## Gender_Male                                                        Gender_Male
## PreferedOrderCat_Mobile                                PreferedOrderCat_Mobile
## PreferredPaymentMode_UPI                              PreferredPaymentMode_UPI
## HourSpendOnApp                                                  HourSpendOnApp
## `PreferredLoginDevice_Mobile Phone`        `PreferredLoginDevice_Mobile Phone`
## PreferedOrderCat_Grocery                              PreferedOrderCat_Grocery
## `PreferredPaymentMode_Debit Card`            `PreferredPaymentMode_Debit Card`
## PreferedOrderCat_Others                                PreferedOrderCat_Others
## `PreferredPaymentMode_Cash on Delivery` `PreferredPaymentMode_Cash on Delivery`
## MaritalStatus_Divorced                                  MaritalStatus_Divorced
##                                              rel.inf
## Tenure                                   53.278762207
## Complain                                 14.600654538
## NumberOfAddress                           6.952966625
## DaySinceLastOrder                         4.806338135
## MaritalStatus_Single                      4.270054928
## NumberOfDeviceRegistered                  2.855939147
## WarehouseToHome                           1.887493685
## CashbackAmount                            1.612901133
## SatisfactionScore                         1.601747566
## `PreferedOrderCat_Laptop & Accessory`     1.205272864
## OrderCount                                1.030088471
## CityTier                                  0.942286059
## `PreferredPaymentMode_E wallet`           0.813682197
## CouponUsed                                0.801774573
## PreferredPaymentMode_COD                  0.715684272
## OrderAmountHikeFromlastYear               0.493724882
## PreferedOrderCat_Fashion                  0.411849720
## PreferredLoginDevice_Computer             0.389015541
## `PreferedOrderCat_Mobile Phone`           0.357908812
## PreferredPaymentMode_CC                   0.261242187
## `PreferredPaymentMode_Credit Card`        0.237733363
## MaritalStatus_Married                     0.225084272
## PreferredLoginDevice_Phone                0.089081462
## Gender_Female                             0.070222006
## Gender_Male                               0.044315360
## PreferedOrderCat_Mobile                   0.012265903
## PreferredPaymentMode_UPI                  0.010271354
## HourSpendOnApp                            0.010189398
## `PreferredLoginDevice_Mobile Phone`       0.005721808
## PreferedOrderCat_Grocery                  0.003042534
## `PreferredPaymentMode_Debit Card`         0.001506346
## PreferedOrderCat_Others                   0.001178654
## `PreferredPaymentMode_Cash on Delivery`   0.000000000
## MaritalStatus_Divorced                    0.000000000
```

```
relative.influence(fit.boost_tree)
```

```
## n.trees not given. Using 10000 trees.
```

```
##                            Tenure                             CityTier
##                      46611.580868                           824.370556
##                   WarehouseToHome                        HourSpendOnApp
##                       1651.297081                             8.914321
##          NumberOfDeviceRegistered                    SatisfactionScore
##                       2498.553514                          1401.308572
##                   NumberOfAddress                             Complain
##                       6082.888429                         12773.562326
##       OrderAmountHikeFromlastYear                           CouponUsed
##                        431.941290                           701.442354
##                        OrderCount                    DaySinceLastOrder
##                        901.185578                          4204.884073
##                    CashbackAmount         PreferredLoginDevice_Computer
##                       1411.066408                           340.335034
##    `PreferredLoginDevice_Mobile Phone`         PreferredLoginDevice_Phone
##                          5.005794                            77.934013
## `PreferredPaymentMode_Cash on Delivery`            PreferredPaymentMode_CC
##                          0.000000                           228.550943
##          PreferredPaymentMode_COD      `PreferredPaymentMode_Credit Card`
##                        626.125194                           207.983958
##      `PreferredPaymentMode_Debit Card`       `PreferredPaymentMode_E wallet`
##                          1.317845                           711.859885
##          PreferredPaymentMode_UPI                        Gender_Female
##                          8.986021                            61.434586
##                       Gender_Male             PreferedOrderCat_Fashion
##                         38.769838                           360.311796
##          PreferedOrderCat_Grocery `PreferedOrderCat_Laptop & Accessory`
##                          2.661798                          1054.447799
##           PreferedOrderCat_Mobile     `PreferedOrderCat_Mobile Phone`
##                         10.730976                           313.120929
##           PreferedOrderCat_Others                MaritalStatus_Divorced
##                          1.031160                             0.000000
##             MaritalStatus_Married                  MaritalStatus_Single
##                        196.917746                          3735.710109
```

Training set MSE

```r
yhat.boost_tree <- predict(fit.boost_tree, train, n.trees = 10000)
mse.boost_tree <- colMeans((yhat.boost_tree - train_y) ^ 2)
print(mse.boost_tree)
```

```
##      Churn
## 0.07583263
```

Testing set MSE

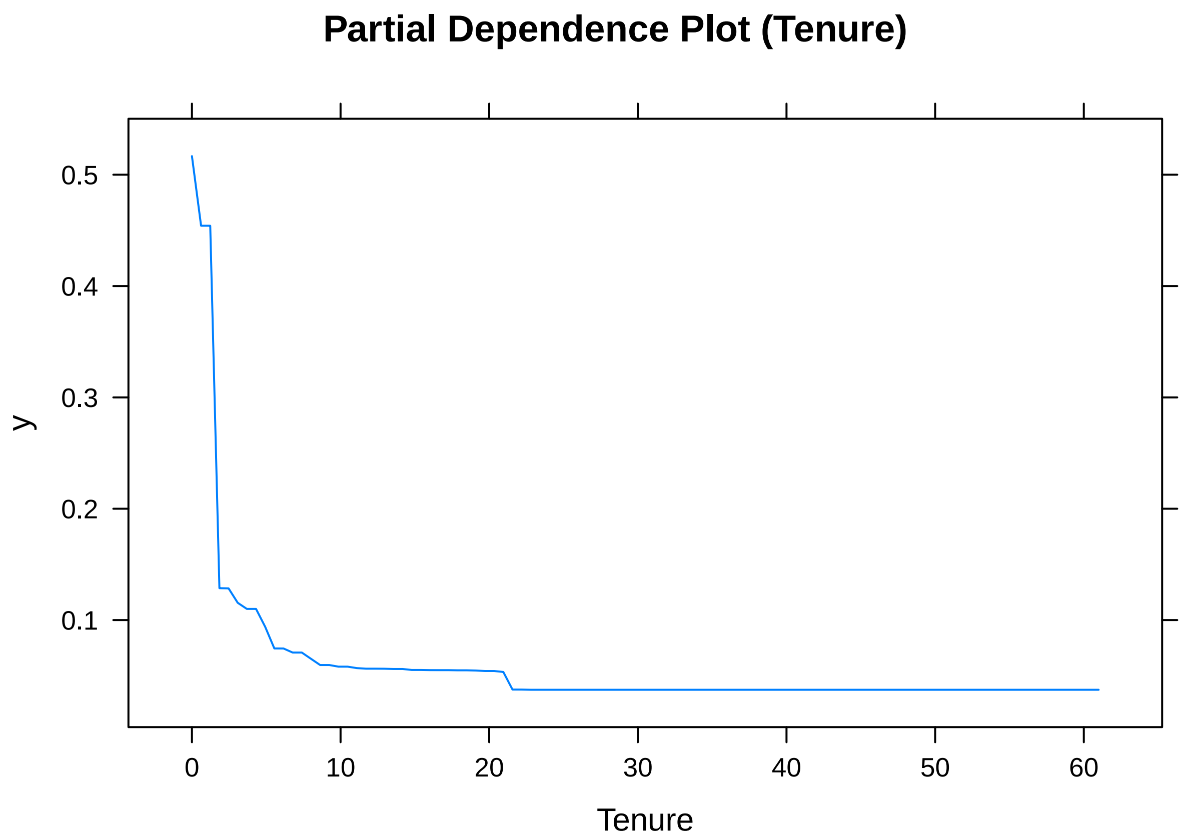
```r
yhat.boost_tree.test <- predict(fit.boost_tree, test, n.trees = 10000)
mse.boost_tree.test <- colMeans((yhat.boost_tree.test - test_y) ^ 2)
print(mse.boost_tree.test)
```

```
##      Churn
## 0.07818863
```

According to the result of the fitted model, the variables 'Tenure' and 'Complain' are the two most important features which explain the maximum variance in the data set.

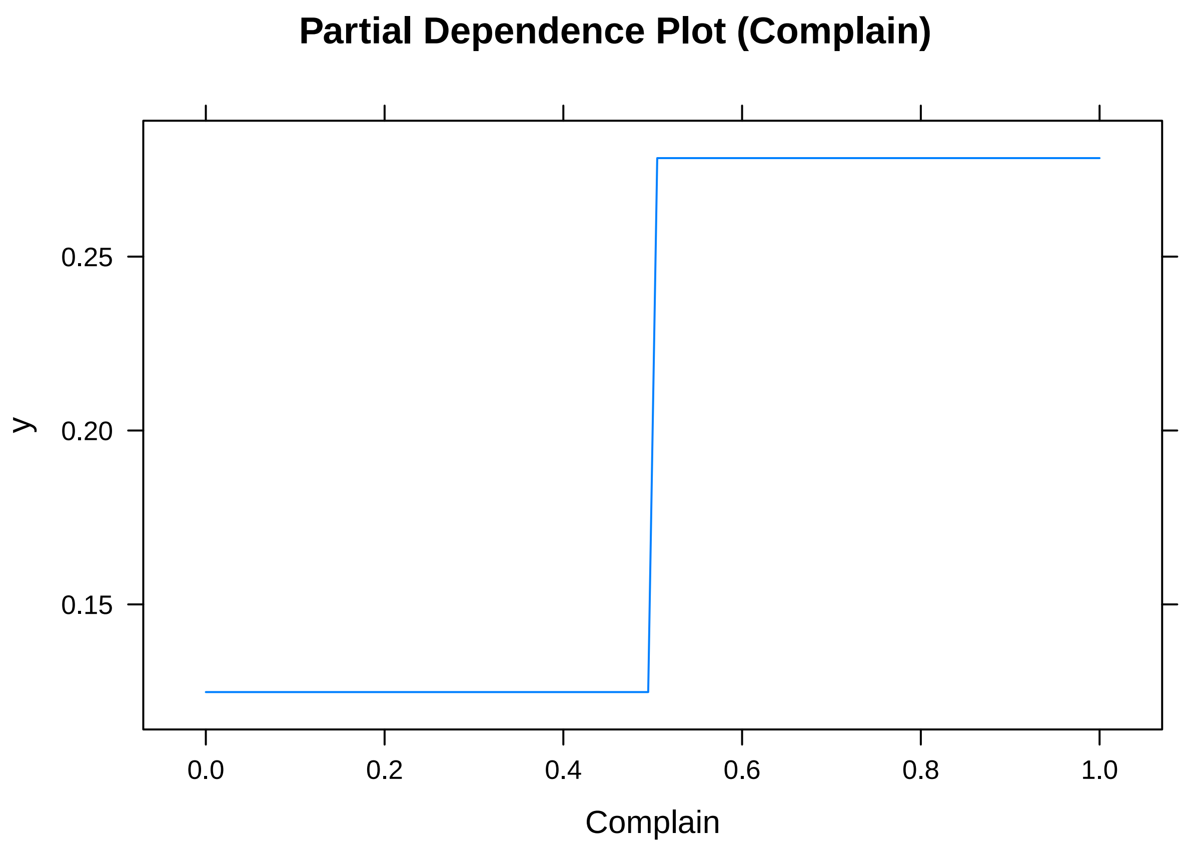
Plotting the partial dependence plot y ~ Tenure

```
plot(fit.boost_tree, i = 'Tenure', main = 'Partial Dependence Plot (Tenure)')
```



y ~ Complain

```
plot(fit.boost_tree, i = 'Complain', main = 'Partial Dependence Plot (Complain)')
```



It turns out that 'Tenure' is negatively correlated with customer churn, while 'Complain' is positively correlated with customer churn. This phenomenon actually makes sense for common situation if 'tenure' of a customer is higher, it shows that the customer has high loyalty. But if 'Complain' = 1, it simply means that the customer is not satisfied with the service, which leads to high probability to churn.

It takes a really long time to run this section. If needed, please un-comment (ctrl + shift + c) this section and run.

```
# # Try to plot the test error vs. number of trees
# Ntree = seq(100, 10000, 100)
#
# train.error = NULL
# test.error = NULL
# for (i in Ntree) {
#   fit.nbtree <- gbm(f1,
#                     data = train,
#                     distribution = "gaussian",
#                     n.trees = i,
#                     interaction.depth = 2,
#                     shrinkage = 0.001)
#
#   train_y_hat_temp <- predict(fit.nbtree, train, n.tree = i)
#   train_mse_temp <- colMeans((train_y_hat_temp - train_y) ^ 2)
#   train.error <- c(train.error, train_mse_temp)
#
#   y_hat_temp <- predict(fit.nbtree, test, n.trees = i)
#   mse_temp <- colMeans((y_hat_temp - test_y) ^ 2)
```

```
#   test.error <- c(test.error, mse_temp)
# }
```

Plotting the test error vs. number of trees

```
# plot(Ntree, test.error, xlab = "Number of Trees", ylab = "Test Error", main = "Perfomance of Boosting
```

We plotted the test MSE vs. number of trees to see if the model is improved when we use higher number of trees to fit the training data. We chose to use ntrees equals 10,000, then the training MSE is 0.0758 and the testing MSE is 0.0782. We do not have a low training MSE with high testing MSE so there is less concern about overfitting.