Churn Prediction classification

Wittawat Muangkot

2022-04-15

This paper will using Churn dataset and do a prediction in classification model

import nescessary package

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

load dataset into R

```
df<- read_csv('churn.csv')
```

```
## Rows: 5000 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): churn, internationalplan, voicemailplan
## dbl (15): accountlength, numbervmailmessages, totaldayminutes, totaldaycalls...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Basic Explore data type of each columns and basic explore dataset

```
sapply(df, class)
```

```
##                     churn              accountlength
##               "character"                  "numeric"
##         internationalplan               voicemailplan
##               "character"                "character"
##        numbervmailmessages            totaldayminutes
##                 "numeric"                  "numeric"
##             totaldaycalls             totaldaycharge
##                 "numeric"                  "numeric"
##           totaleveminutes              totalevecalls
##                 "numeric"                  "numeric"
##            totalevecharge          totalnightminutes
##                 "numeric"                  "numeric"
##           totalnightcalls           totalnightcharge
##                 "numeric"                  "numeric"
##          totalintlminutes             totalintlcalls
##                 "numeric"                  "numeric"
##           totalintlcharge numbercustomerservicecalls
##                 "numeric"                  "numeric"
```

```
df%>%
  glimpse()
```

```
## Rows: 5,000
## Columns: 18
## $ churn                      <chr> "No", "No", "No", "No", "No", "No", "No", "~
## $ accountlength              <dbl> 128, 107, 137, 84, 75, 118, 121, 147, 117, ~
## $ internationalplan          <chr> "no", "no", "no", "yes", "yes", "yes", "no"~
## $ voicemailplan              <chr> "yes", "yes", "no", "no", "no", "no", "yes"~
## $ numbervmailmessages        <dbl> 25, 26, 0, 0, 0, 0, 24, 0, 0, 37, 0, 0, 0, ~
## $ totaldayminutes            <dbl> 265.1, 161.6, 243.4, 299.4, 166.7, 223.4, 2~
## $ totaldaycalls              <dbl> 110, 123, 114, 71, 113, 98, 88, 79, 97, 84,~
## $ totaldaycharge             <dbl> 45.07, 27.47, 41.38, 50.90, 28.34, 37.98, 3~
## $ totaleveminutes            <dbl> 197.4, 195.5, 121.2, 61.9, 148.3, 220.6, 34~
## $ totalevecalls              <dbl> 99, 103, 110, 88, 122, 101, 108, 94, 80, 11~
## $ totalevecharge             <dbl> 16.78, 16.62, 10.30, 5.26, 12.61, 18.75, 29~
## $ totalnightminutes          <dbl> 244.7, 254.4, 162.6, 196.9, 186.9, 203.9, 2~
## $ totalnightcalls            <dbl> 91, 103, 104, 89, 121, 118, 118, 96, 90, 97~
## $ totalnightcharge           <dbl> 11.01, 11.45, 7.32, 8.86, 8.41, 9.18, 9.57,~
## $ totalintlminutes           <dbl> 10.0, 13.7, 12.2, 6.6, 10.1, 6.3, 7.5, 7.1,~
## $ totalintlcalls             <dbl> 3, 3, 5, 7, 3, 6, 7, 6, 4, 5, 6, 5, 2, 5, 6~
## $ totalintlcharge            <dbl> 2.70, 3.70, 3.29, 1.78, 2.73, 1.70, 2.03, 1~
## $ numbercustomerservicecalls <dbl> 1, 1, 0, 2, 3, 0, 3, 0, 1, 0, 4, 0, 1, 3, 4~
```

Transfrom Category columns into factor

```
df<- df%>%
  mutate_if(is.character,as.factor)
```

Check missing values

```
mean(complete.cases(df))
```

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

check base line prediction

to check base line prediction by count values, notice that No churn has 85.86 percent. so with no model and only predict no can get result 85% from this dataset

```
df%>%
  count(churn)%>%
  mutate(percent=n/sum(n))
```

```
## # A tibble: 2 x 3
##   churn     n percent
##   <fct> <int>   <dbl>
## 1 No     4293   0.859
## 2 Yes     707   0.141
```

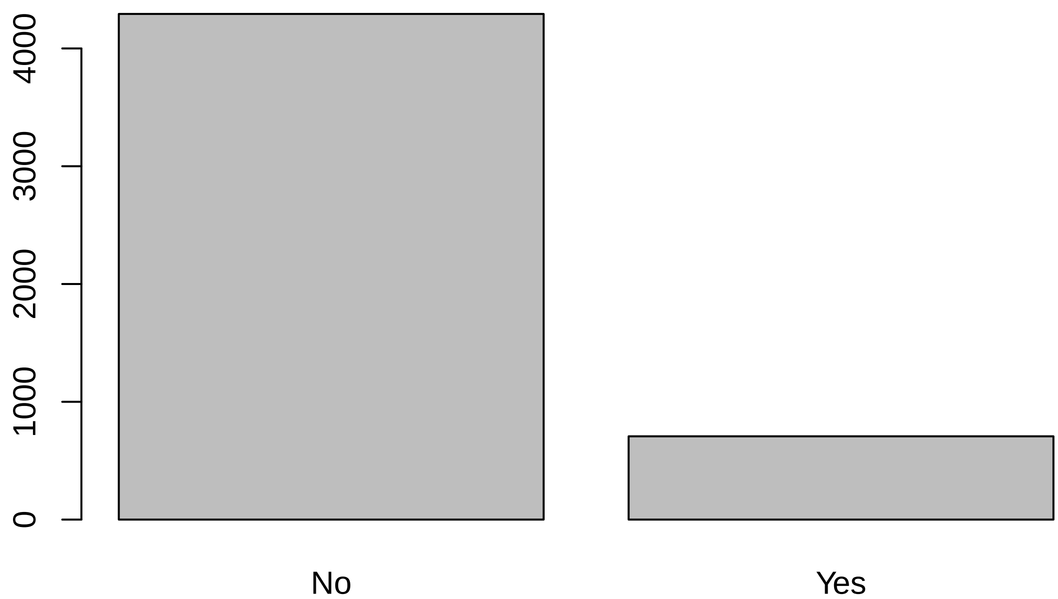
Train test split data using train size 80%

```
set.seed(42)
id<- createDataPartition(y=df$churn,
                         p = 0.8,
                         list = FALSE)
```

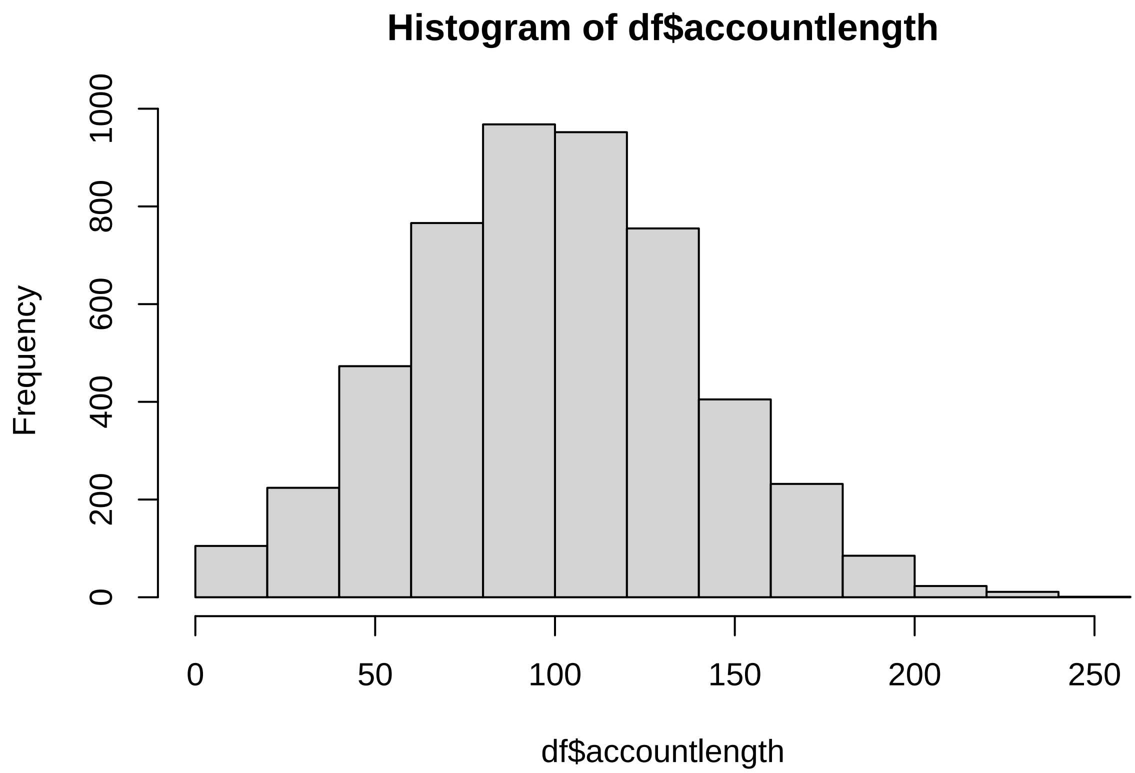
```
train<- df[id, ]
test<- df[-id, ]
```

Basic Visualization

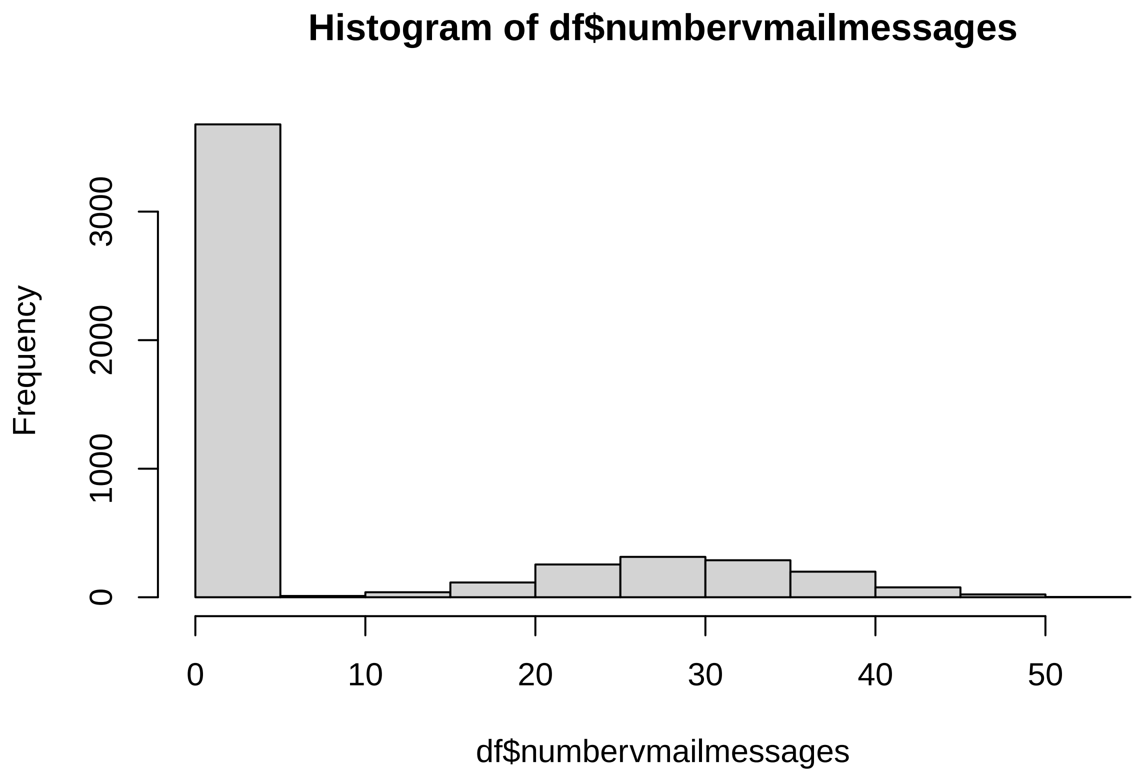
```
plot(df$churn)
```



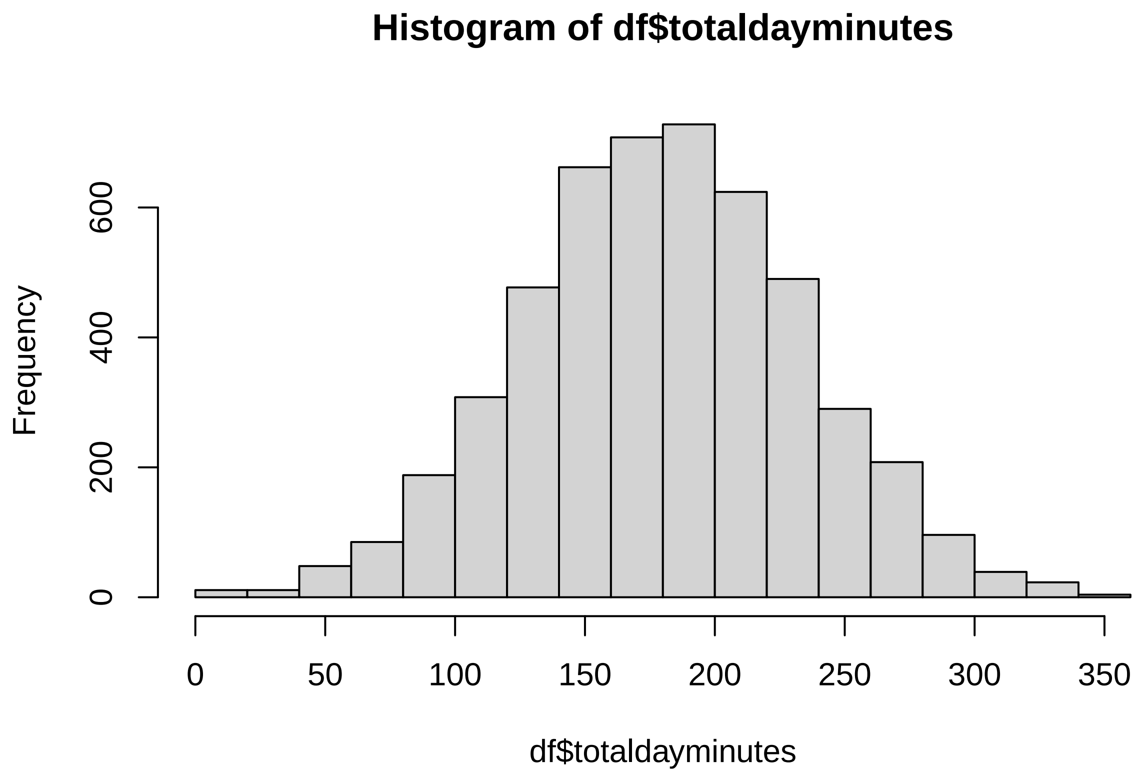

```
hist(df$accountlength)
```



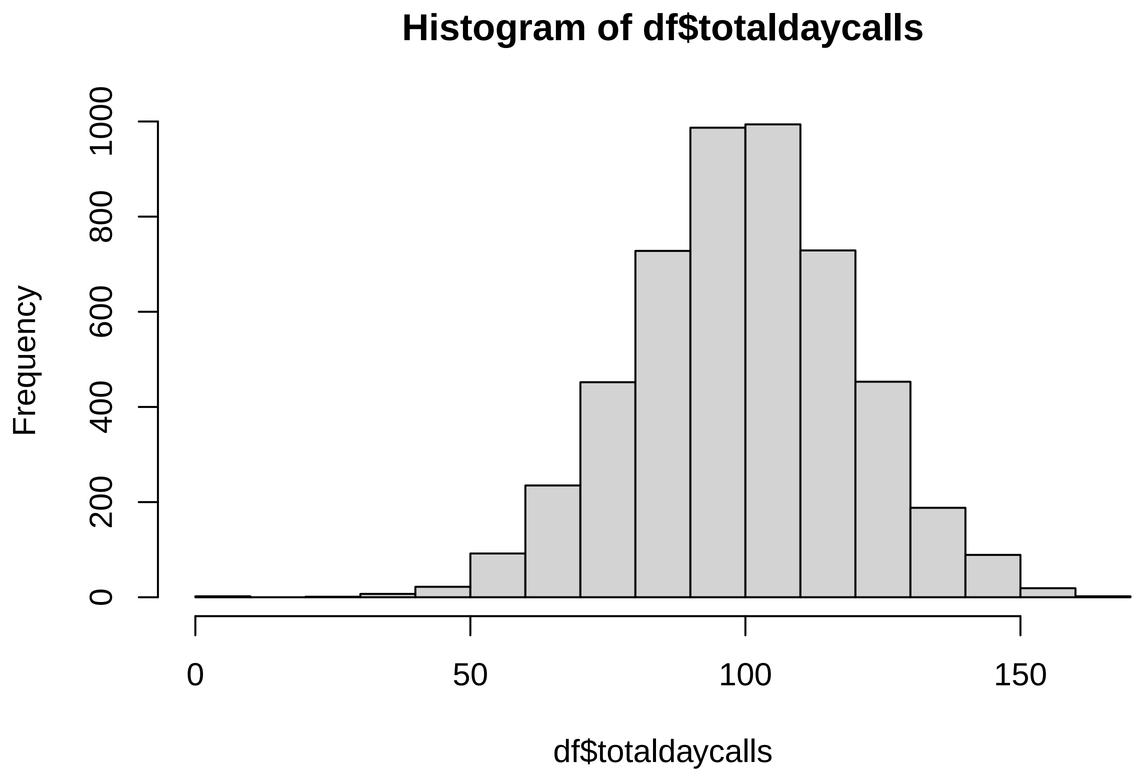

```
hist(df$numbervmailmessages)
```



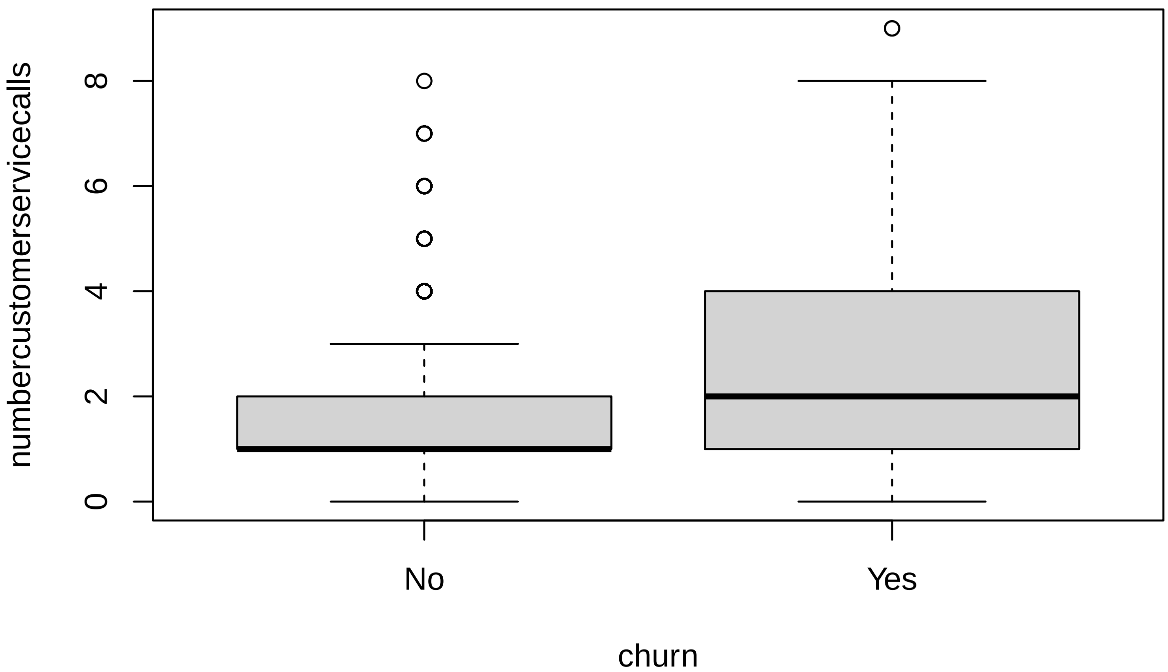

```
hist(df$totaldayminutes)
```



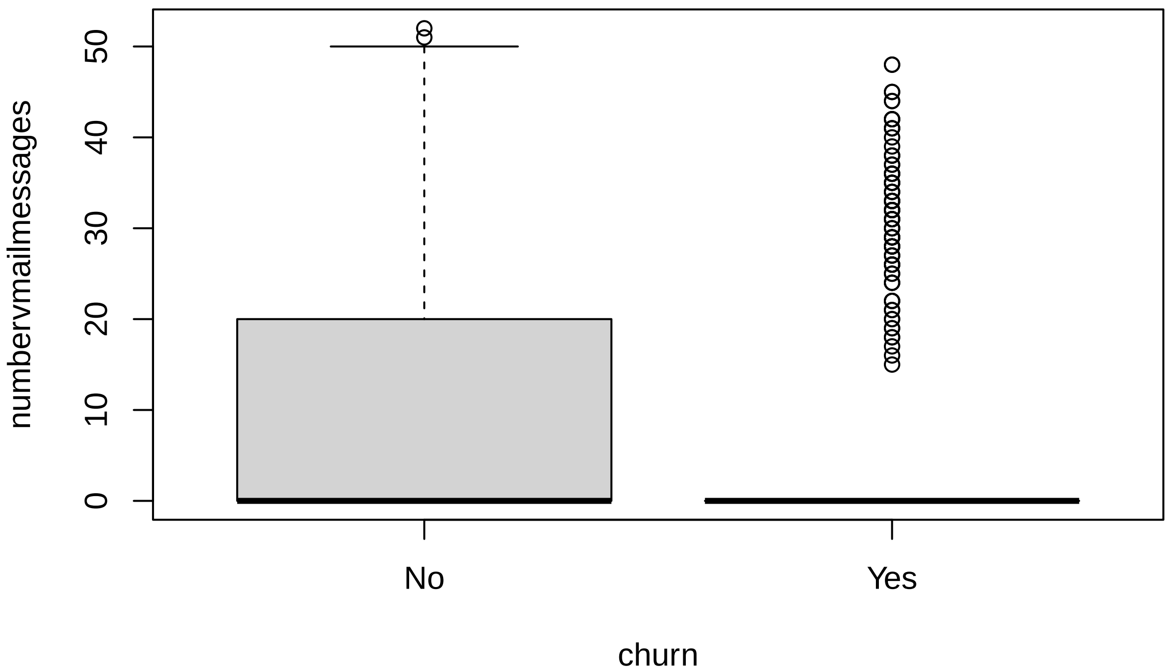

```
hist(df$totaldaycalls )
```



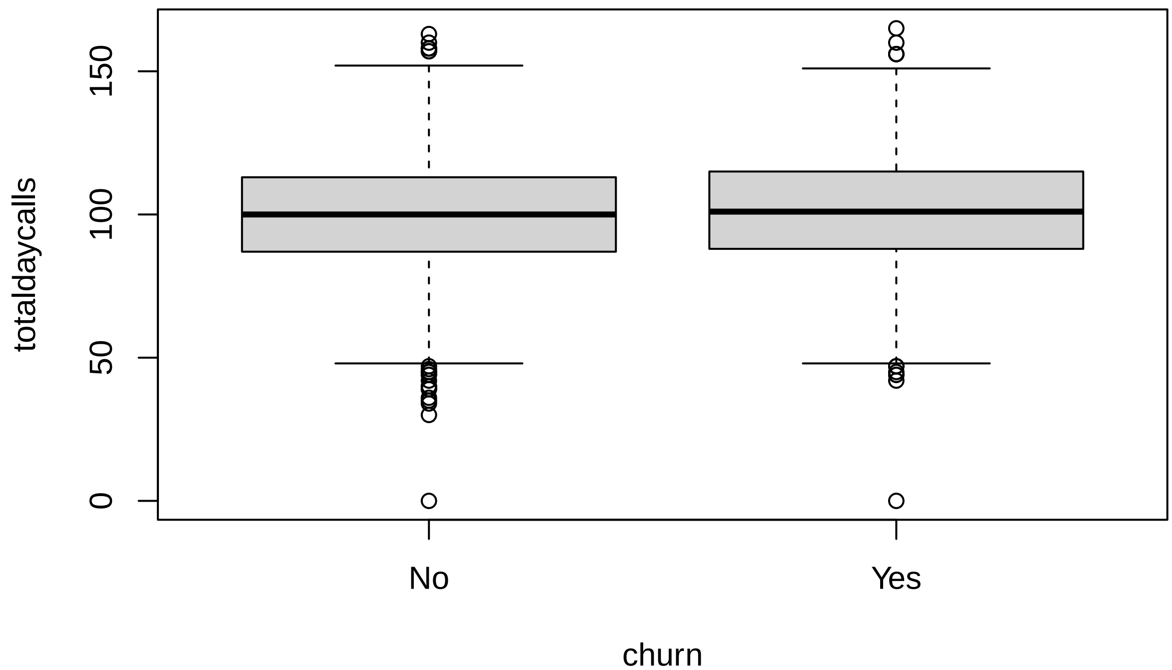

```
boxplot(numbercustomerservicecalls ~churn,data = df)
```



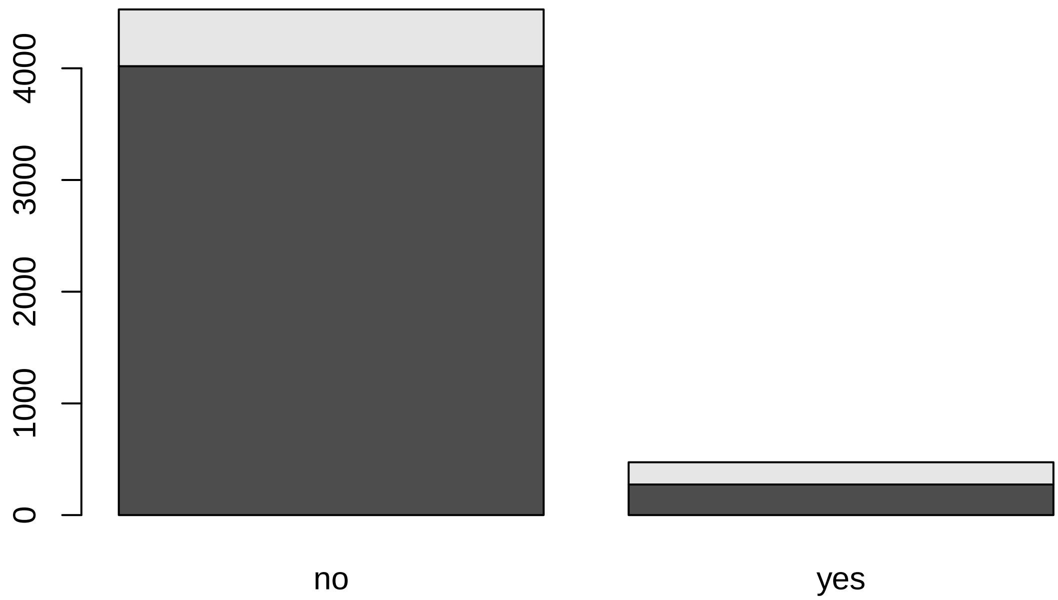

```r
boxplot(numbervmailmessages~churn,data = df)
```



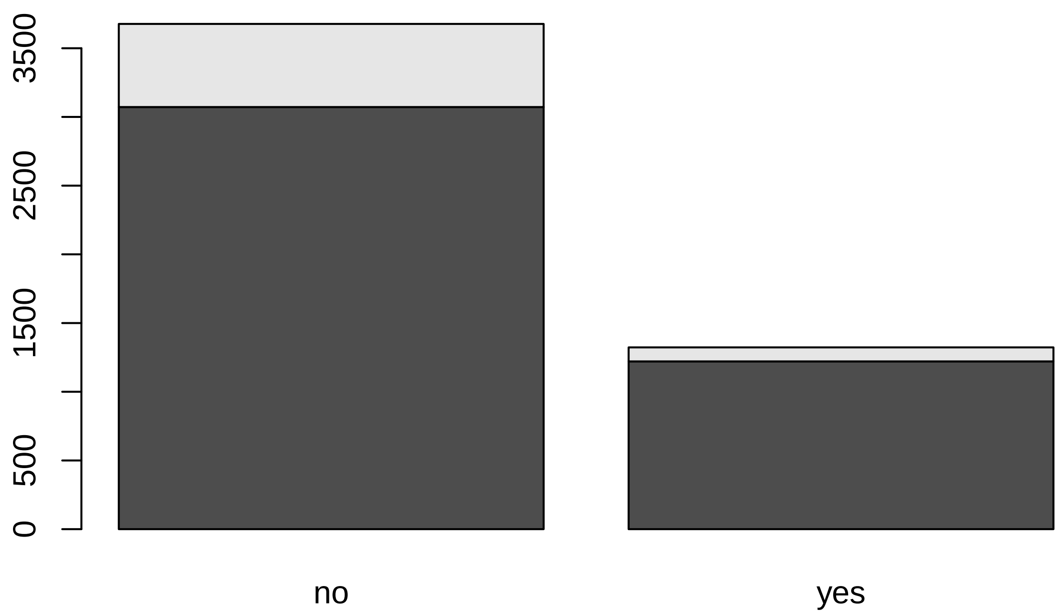

```
boxplot(totaldaycalls ~churn,data = df)
```




```
barplot(table(df$churn,df$internationalplan, dnn = c('Churn','Interplan')))
```




```r
barplot(table(df$churn,df$voicemailplan, dnn = c('Churn','voicemailplan')))
```



model selection

to find best algorithm by compare model randomforest, logistic regression and k nearest neighbor

```
ctrl<- trainControl(method = 'none',
                      classProbs = TRUE,
                      summaryFunction = twoClassSummary,
                      verboseIter = TRUE)

model<- train(churn ~.,
                data=train,
                method= 'rf',
                metric= 'ROC',
                trControl=ctrl)
```

```
## Fitting mtry = 4 on full training set
```

```
pred<- predict(model,newdata = test)
```

```
accuracy<- mean(pred == test$churn)
accuracy
```

```
## [1] 0.963964
```

```
ctrl<- trainControl(method = 'none',
                    classProbs = TRUE,
                    summaryFunction = twoClassSummary,
                    verboseIter = TRUE)

model<- train(churn ~.,
              data=train,
              method= 'glm',
              metric= 'ROC',
              trControl=ctrl)
```

```
## Fitting parameter = none on full training set
```

```
pred<- predict(model,newdata = test)

accuracy<- mean(pred == test$churn)
accuracy
```

```
## [1] 0.8598599
```

```
ctrl<- trainControl(method = 'none',
                    classProbs = TRUE,
                    summaryFunction = twoClassSummary,
                    verboseIter = TRUE)

model<- train(churn ~.,
              data=train,
              method= 'knn',
              metric= 'ROC',
              trControl=ctrl)
```

```
## Fitting k = 5 on full training set
```

```
pred<- predict(model,newdata = test)

accuracy<- mean(pred == test$churn)
accuracy
```

```
## [1] 0.8878879
```

The best accuracy from 3 model that we fit to this dataset is the random forest

the Feature Engineering to increase model accuracy

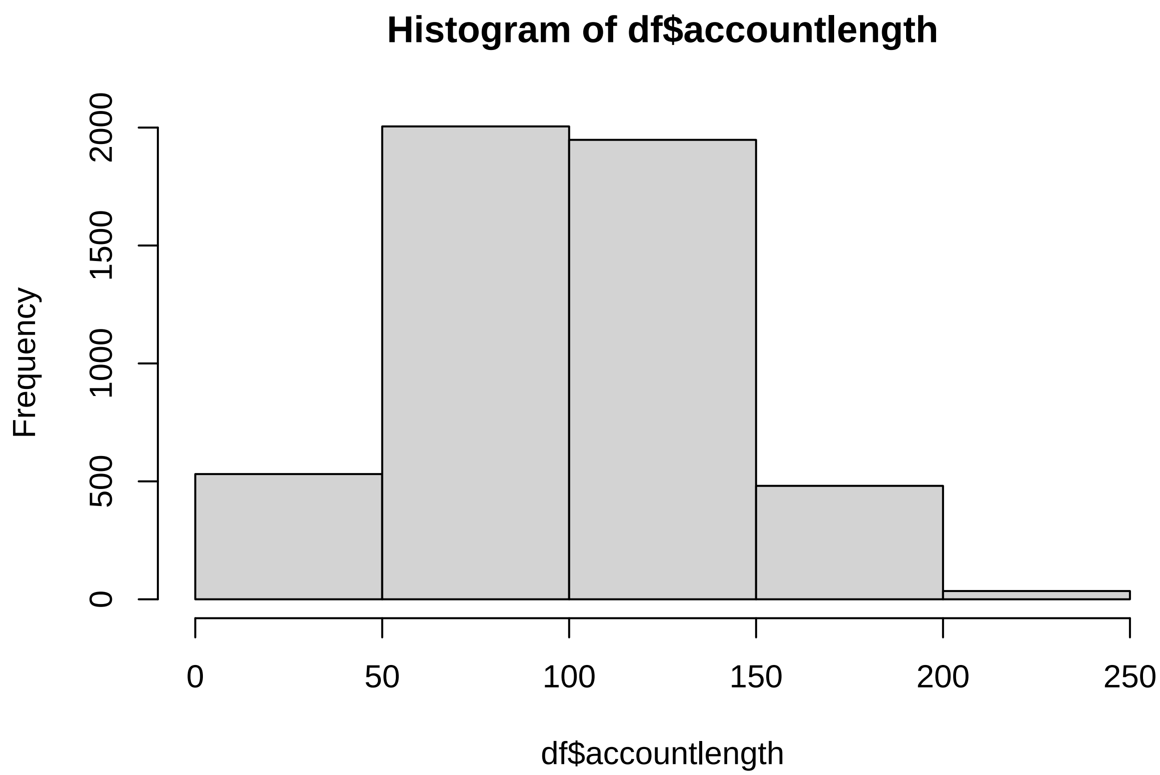
```
df<- df%>%
  mutate(totalcall= totaldaycalls+totalevecalls +totalnightcalls+totalintlcalls)

df<- df%>%
```

```
  mutate(totalminutes= totaldayminutes+totaleveminutes+totalnightminutes+totalintlminutes)

df<- df%>%
  mutate(totalcharge= totaldaycharge+totalevecharge+totalnightcharge+totalintlcharge)

df<- df%>%
  mutate(ratiodaycall= totaldaycalls/totalcall)
df<- df%>%
  mutate(ratioevecall= totalevecalls/totalcall)
df<- df%>%
  mutate(rationightcall= totalnightcalls/totalcall)
df<- df%>%
  mutate(ratiointlcall= totalintlcalls/totalcall)

df<- df%>%
  mutate(ratiodayminutes= totaldayminutes/totalminutes)
df<- df%>%
  mutate(ratioeveminutes= totaleveminutes/totalminutes)
df<- df%>%
  mutate(rationightminutes= totalnightminutes/totalminutes)
df<- df%>%
  mutate(ratiointlminutes= totalintlminutes/totalminutes)
df<- df%>%
  mutate(ratiodaycharge= totaldaycharge/totalcharge)
df<- df%>%
  mutate(ratioevecharge= totalevecharge/totalcharge)
df<- df%>%
  mutate(rationightcharge= totalnightcharge/totalcharge)
df<- df%>%
  mutate(ratiointlcharge= totalintlcharge/totalcharge)

df<-df%>%
  group_by(internationalplan)%>%
  mutate(Avgchargebyinterplan= mean(totalcharge))

df<-df%>%
  group_by(internationalplan)%>%
  mutate(Avgminutesbyinterplan= mean(totalminutes))

df<-df%>%
  group_by(internationalplan)%>%
  mutate(Avgcallbyinterplan= mean(totalcall))


df<-df%>%
  group_by(voicemailplan )%>%
  mutate(Avgchargebyvoicemailplan = mean(totalcharge))

df<-df%>%
  group_by(voicemailplan )%>%
  mutate(Avgminutesbyvoicemailplan = mean(totalminutes))

df<-df%>%
```

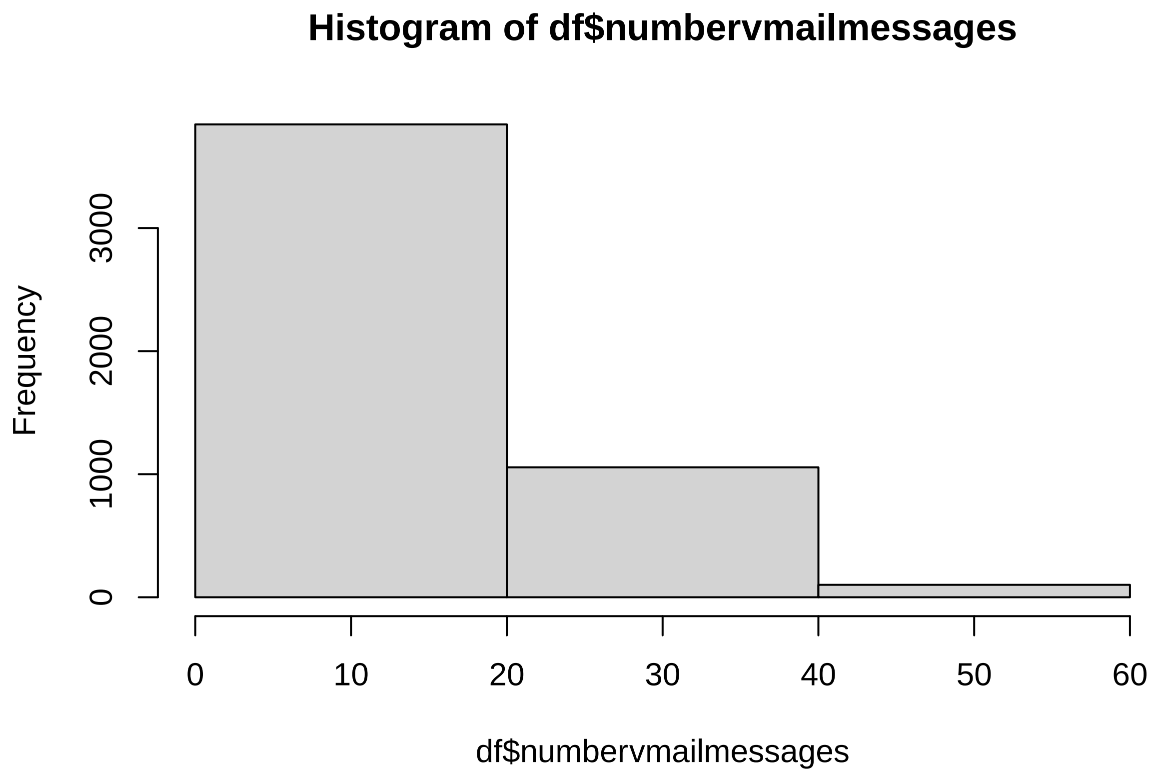
```
  group_by(voicemailplan )%>%
  mutate(Avgcallbyvoicemailplan = mean(totalcall))
```

visualize to find the best number of bins in each chart

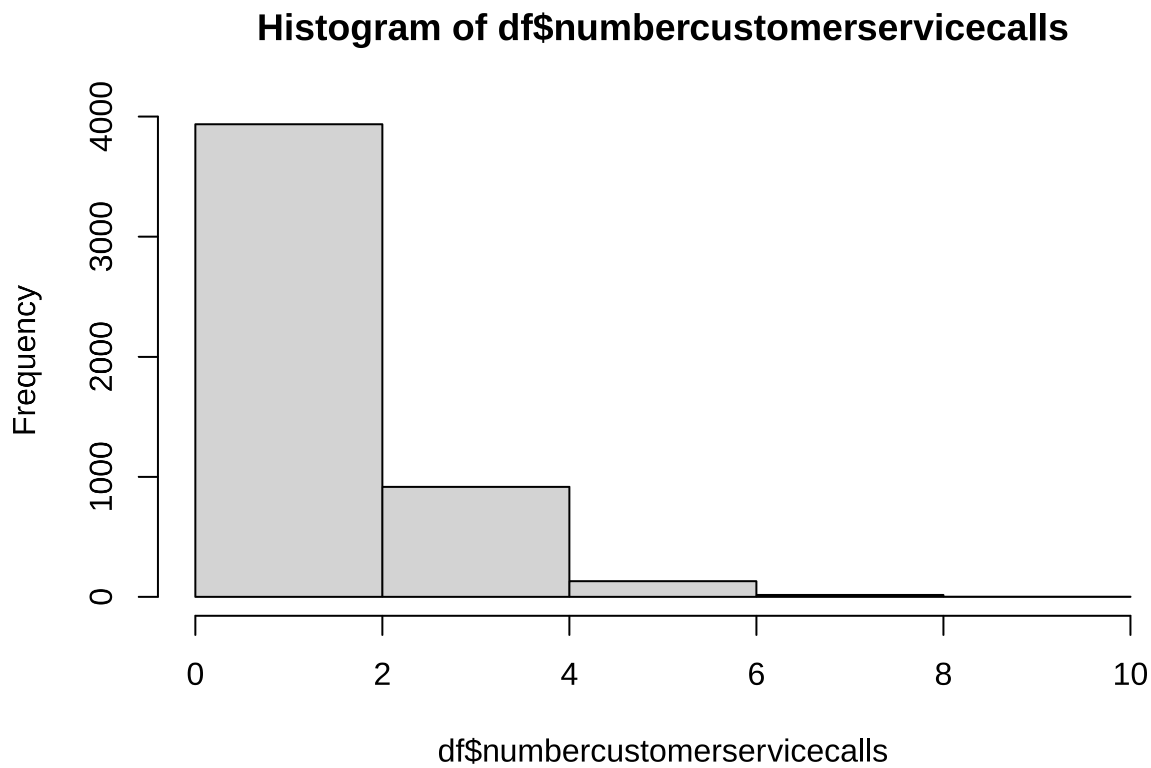
```
hist(df$accountlength,4)
```




```
hist(df$numbervmailmessages,3)
```




```
hist(df$numbercustomerservicecalls,4)
```



descrete data into group refer bins numbers

```
df<-df%>%
  mutate(Groupaccountlength = cut(accountlength,4,labels=FALSE))

df<-df%>%
  mutate(Groupnumbervmailmessages = cut(numbervmailmessages,3,labels=FALSE))

df<-df%>%
  mutate(Groupnumbercustomerservicecalls = cut(numbercustomerservicecalls,4,labels=FALSE))
```

split dataset into train and test for evaluate model

```
set.seed(42)
id<- createDataPartition(y=df$churn,
                         p = 0.8,
                         list = FALSE)
train<- df[id, ]
test<- df[-id, ]
```

fit train dataset into randomforest model

using resample method k fold cross validation and separated data into 5 preprocess using scale for scale dataset into same range and using YeoJohnson method to transform dataset into normal distribution and using ROC matrix for evaluate model

```
set.seed(42)

ctrl<- trainControl(method = 'cv',
                    number=5,

                    classProbs = TRUE,
                    summaryFunction = twoClassSummary,
                    verboseIter = TRUE)

rf<- train(churn ~.,
      data=train,
      method= 'rf',
      metric= 'ROC',
      trControl=ctrl,
      preProcess= c("YeoJohnson","scale"))
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=21
## - Fold1: mtry=21
## + Fold1: mtry=41
## - Fold1: mtry=41
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=21
## - Fold2: mtry=21
## + Fold2: mtry=41
## - Fold2: mtry=41
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=21
## - Fold3: mtry=21
## + Fold3: mtry=41
## - Fold3: mtry=41
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=21
## - Fold4: mtry=21
## + Fold4: mtry=41
## - Fold4: mtry=41
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=21
## - Fold5: mtry=21
## + Fold5: mtry=41
## - Fold5: mtry=41
## Aggregating results
## Selecting tuning parameters
```

```
## Fitting mtry = 21 on full training set
```

```r
rf
```

```
## Random Forest
##
## 4001 samples
##   41 predictor
##    2 classes: 'No', 'Yes'
##
## Pre-processing: Yeo-Johnson transformation (41), scaled (41)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3201, 3201, 3201, 3201, 3200
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    2    0.9219343  0.9994178  0.6200590
##   21    0.9300002  0.9991266  0.8568079
##   41    0.9274816  0.9988355  0.8515137
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 21.
```

the best parameter for the best ROC is use mtry = 43

prediction

```r
pred<- predict(rf,newdata = test)
```

evaluate model

```r
accuracy<- mean(pred == test$churn)
accuracy
```

```
## [1] 0.974975
```

after do the feature engineering and find the best parameter, we got the accuracy 97.5 %

after find the best model for this dataset then we calculate the confusion matrix

```r
conf<- confusionMatrix(pred,test$churn,
                       mode= 'prec_recall',
                       positive = 'Yes' )

conf
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  No Yes
##        No  857  24
##        Yes   1 117
##
##                Accuracy : 0.975
##                  95% CI : (0.9633, 0.9837)
##     No Information Rate : 0.8589
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8892
##
##  Mcnemar's Test P-Value : 1.083e-05
##
##               Precision : 0.9915
##                  Recall : 0.8298
##                      F1 : 0.9035
##              Prevalence : 0.1411
##          Detection Rate : 0.1171
##    Detection Prevalence : 0.1181
##       Balanced Accuracy : 0.9143
##
##        'Positive' Class : Yes
##
```

Conclusion

As this model we try to predict the customer churn rate, so we will focus the predict yes and set the Positive Class to Yes and the result of accuracy got 97.5 %, at confidence interval 95% got range 96.33% to 98.37, As the P-values <0.05 show that the overall model is significance. For Precision show that all prediction with churn has high rate prediction correction 99.15%, only 0.85% mising prediction. For Racall show that the total actual churn the model can predict right 82.98%. and the F1 score the model got 90.35%.

save and load model for model deployment

```r
saveRDS(rf, 'rf_churn.RDS')


model<- readRDS('rf_churn.RDS')
```