Random Forest

Tianshu Zhao

November 26, 2018

Data Implement

```
df2 <- read.csv("C:/Users/ariel/Dropbox/Stat154/Project/data/newdata21.csv")
```

0 is bad loan, 1 is good loan, 2 is current loan status Perform regression tree on the data set:

```
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
```

```
random <- sample(80000,500)
new.df <- df2[random,]
```

Find the non current loan

```
non_current_df <- new.df[new.df$loan_rate<2,]
```

```
library("tree")
set.seed(1)
bad <- ifelse(non_current_df$loan_rate <= 0,"Bad","Good")
```

```
non_current_df$term <- NULL
non_current_df$grade <- NULL
non_current_df$sub_grade <- NULL
non_current_df$addr_state <- NULL
non_current_df$initial_list_status <- NULL
```

Random Forest

```
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:dplyr':
##
##     combine
```

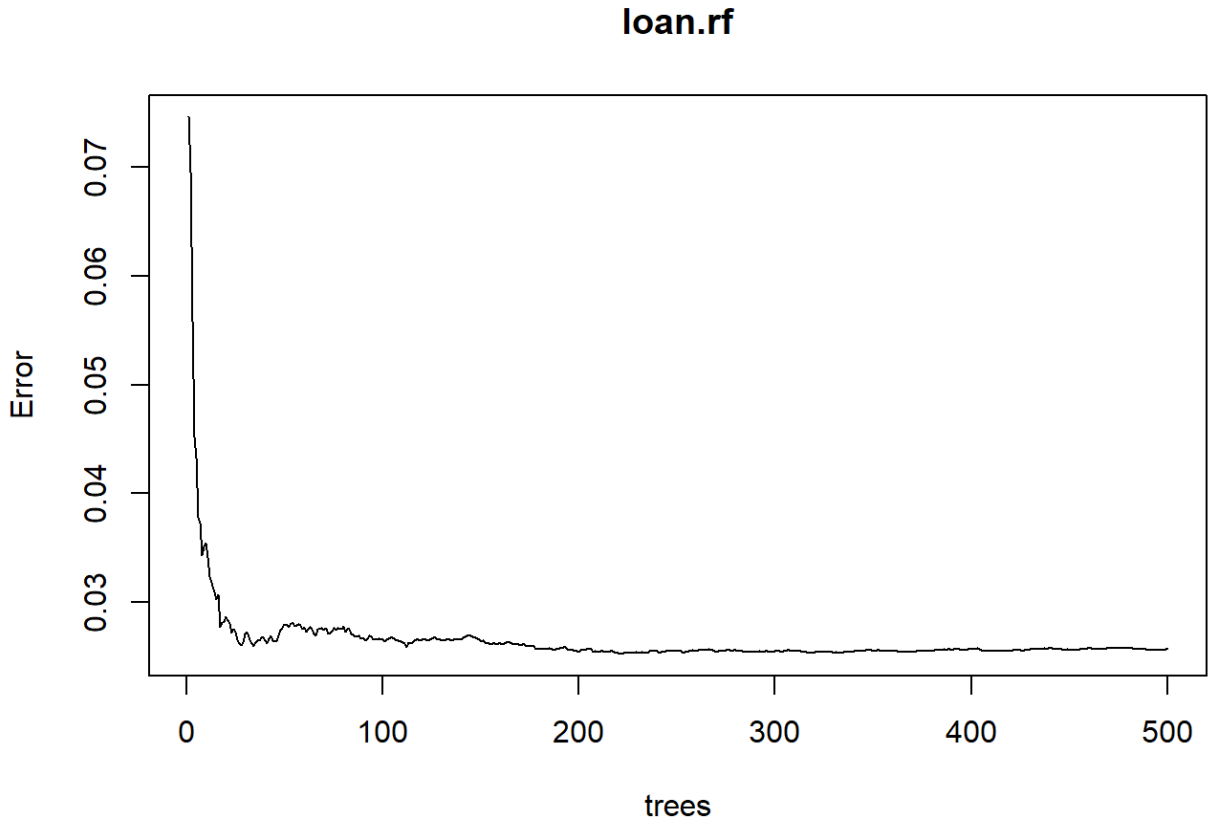
```
loan.rf <- randomForest(loan_rate~.,data = non_current_df,keep.forest=TRUE)
```

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

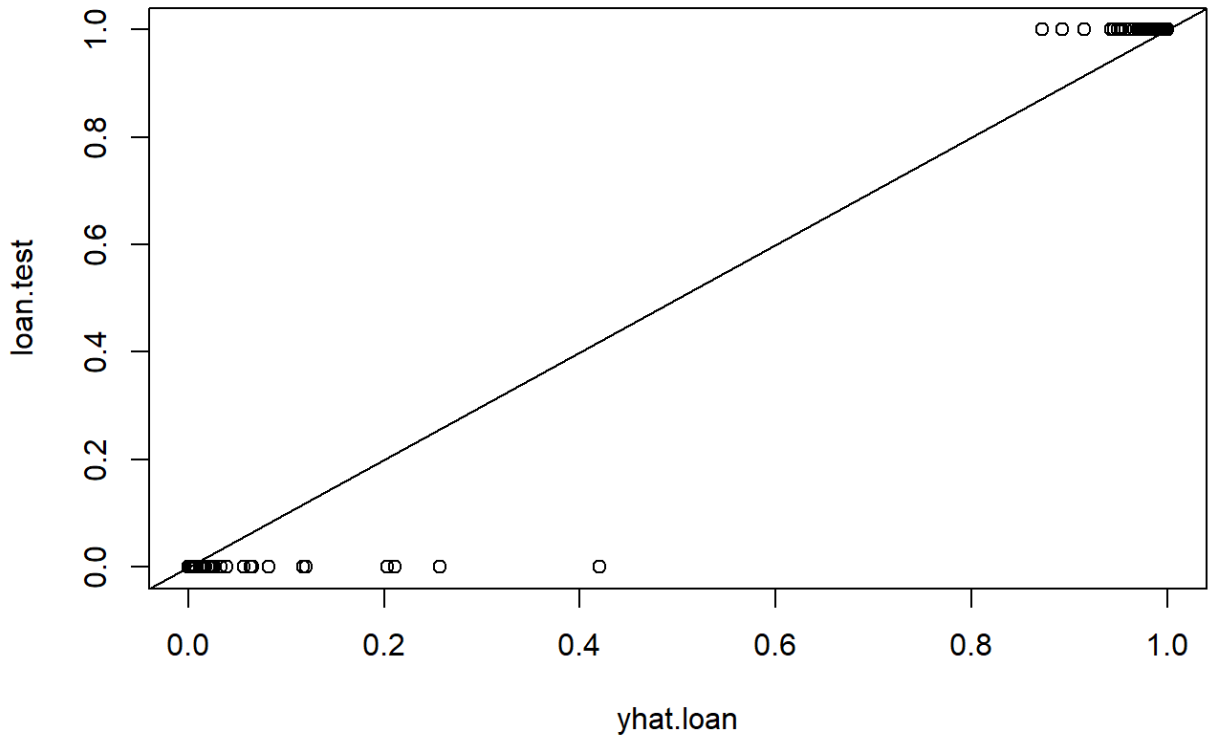
```
loan.rf
```

```
##
## Call:
##  randomForest(formula = loan_rate ~ ., data = non_current_df,      keep.forest = TR
UE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##           Mean of squared residuals: 0.02569284
##                     % Var explained: 85.03
```

```
plot(loan.rf)
```




```
train = sample(1:nrow(non_current_df),nrow(non_current_df)/2) # randomly divide the da
taset into a training set and a test set
loan.test = non_current_df[-train,"loan_rate"]
yhat.loan = predict(loan.rf,newdata = non_current_df[-train,])
plot(yhat.loan,loan.test)
abline(0,1)
```



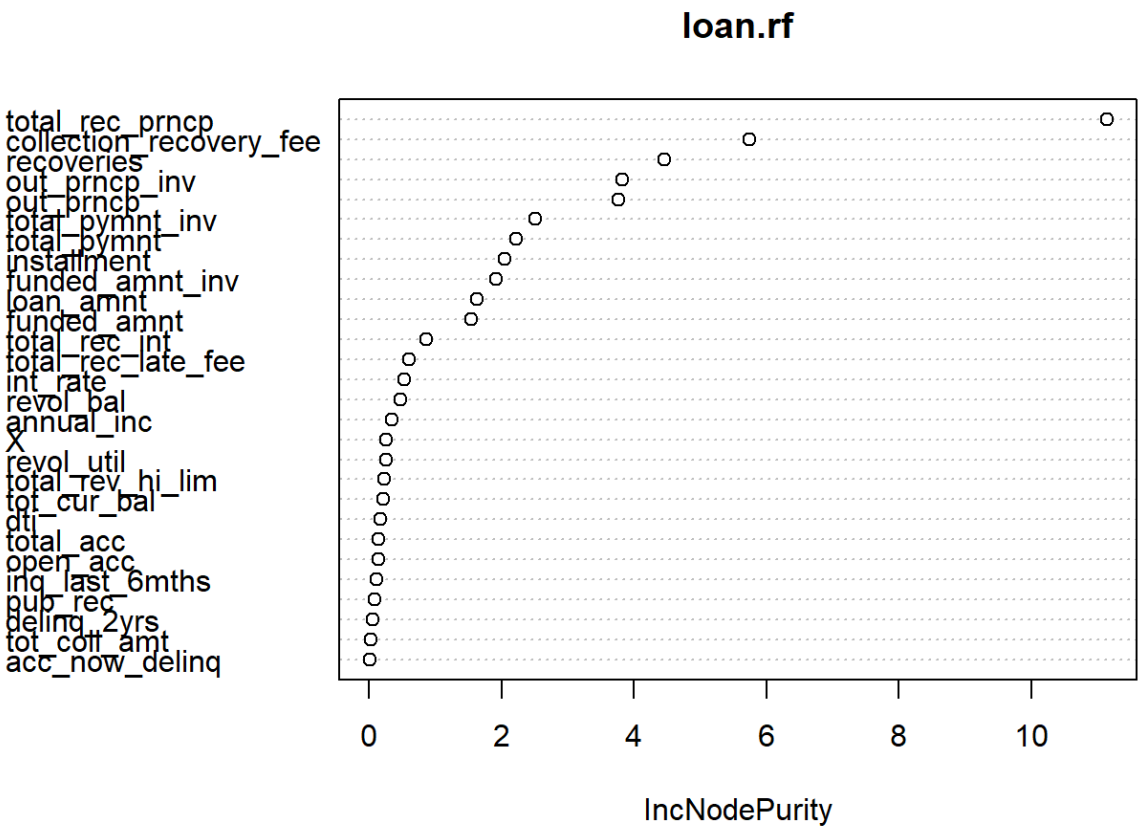

```
mean((yhat.loan-loan.test)^2)
```

```
## [1] 0.003325449
```

```
importance(loan.rf)
```

```
##                         IncNodePurity
## X                          0.25807920
## loan_amnt                  1.63289205
## funded_amnt                1.54277058
## funded_amnt_inv            1.91350808
## int_rate                   0.53159346
## installment                2.05336579
## annual_inc                 0.34186249
## dti                        0.17287560
## delinq_2yrs                0.05275026
## inq_last_6mths             0.11844738
## open_acc                   0.13600779
## pub_rec                    0.08725264
## revol_bal                  0.46869459
## revol_util                 0.25280400
## total_acc                  0.13970480
## out_prncp                  3.76590831
## out_prncp_inv              3.81774855
## total_pymnt                2.22534574
## total_pymnt_inv            2.51461258
## total_rec_prncp           11.13906040
## total_rec_int              0.86624811
## total_rec_late_fee         0.61022287
## recoveries                 4.45348261
## collection_recovery_fee    5.73425579
## acc_now_delinq             0.00503206
## tot_coll_amt               0.02000893
## tot_cur_bal                0.21554795
## total_rev_hi_lim           0.23336167
```

```
varImpPlot(loan.rf)
```


loan.rf
total_rec_prncp
collection_recovery_fee
recoveries
out_prncp_inv
out_prncp
total_pymnt_inv
total_pymnt
installment
funded_amnt_inv
loan_amnt
funded_amnt
total_rec_int
total_rec_late_fee
int_rate
revol_bal
annual_inc
X
revol_util
total_rev_hi_lim
tot_cur_bal
dti
total_acc
open_acc
inq_last_6mths
pub_rec
delinq_2yrs
tot_coll_amt
acc_now_delinq
0
2
4
6
8
10
IncNodePurity