data_challenge

Zongyan Wang

March 25, 2016

Read data and packages

```r
# Upload packages for data clean and analysis
require(plyr)  #data clean
```

```
## Loading required package: plyr
```

```r
require(dplyr) #data clean
```

```
## Loading required package: dplyr
```

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

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

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

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

```r
require(tidyr) #data clean
```

```
## Loading required package: tidyr
```

```r
require(reshape2) # data clean
```

```
## Loading required package: reshape2
```

```r
require(ggplot2)  # visualzation
```

```
## Loading required package: ggplot2
```

```r
require(XML)   # web scriping
```

```
## Loading required package: XML
```

```r
require(testthat) #test model
```

```
## Loading required package: testthat
```

```r
require(kernlab) # kernel and SVM
```

```
## Loading required package: kernlab

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
require(datasets) # State data
require(caret) # Cross - Validation k fold
```

```
## Loading required package: caret

## Loading required package: lattice
```

```r
require(RCurl) # load the website link
```

```
## Loading required package: RCurl

## Loading required package: bitops

##
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':
##
##     complete
```

```r
require(maps) # heatmap: map_data
```

```
## Loading required package: maps

##
##  # maps v3.1: updated 'world': all lakes moved to separate new #
##  # 'lakes' database. Type '?world' or 'news(package="maps")'.  #
```

```
##
## Attaching package: 'maps'

## The following object is masked from 'package:plyr':
##
##     ozone
```

```r
require(ggmap) # for heatmap
```

```
## Loading required package: ggmap
```

```r
require(gridExtra)
```

```
## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

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

Read file

```r
# Read Data and function files
LC <- read.csv(file.choose(), header = T) #Please read LC_biz_all.csv
source("clean_f2.R") # functions for cleaning the data
source("plot_f2.R") # functions for plot(heatmap)
source("analysis_f2.R") # functions for plot and data analysis
```

Data clean

```r
lc <- perToN(LC)  # transfer percentage to numeric
lc <- replaceBlank_all(lc) # replace the blank
# Transform earliest_cr_line, issue_d, last_pymnt_d from str to numeric
lc <- lc %>% dateToNum(., "earliest_cr_line") %>%
  dateToNum(., "issue_d") %>% dateToNum(., "last_pymnt_d")

lc <- lc %>% select(-id,  -emp_title,
                    -last_pymnt_d, -next_pymnt_d, -last_pymnt_amnt,
                    -zip_code,-issue_d, -last_credit_pull_d)
```

Data Summary

```
n.factor_all(lc)
```

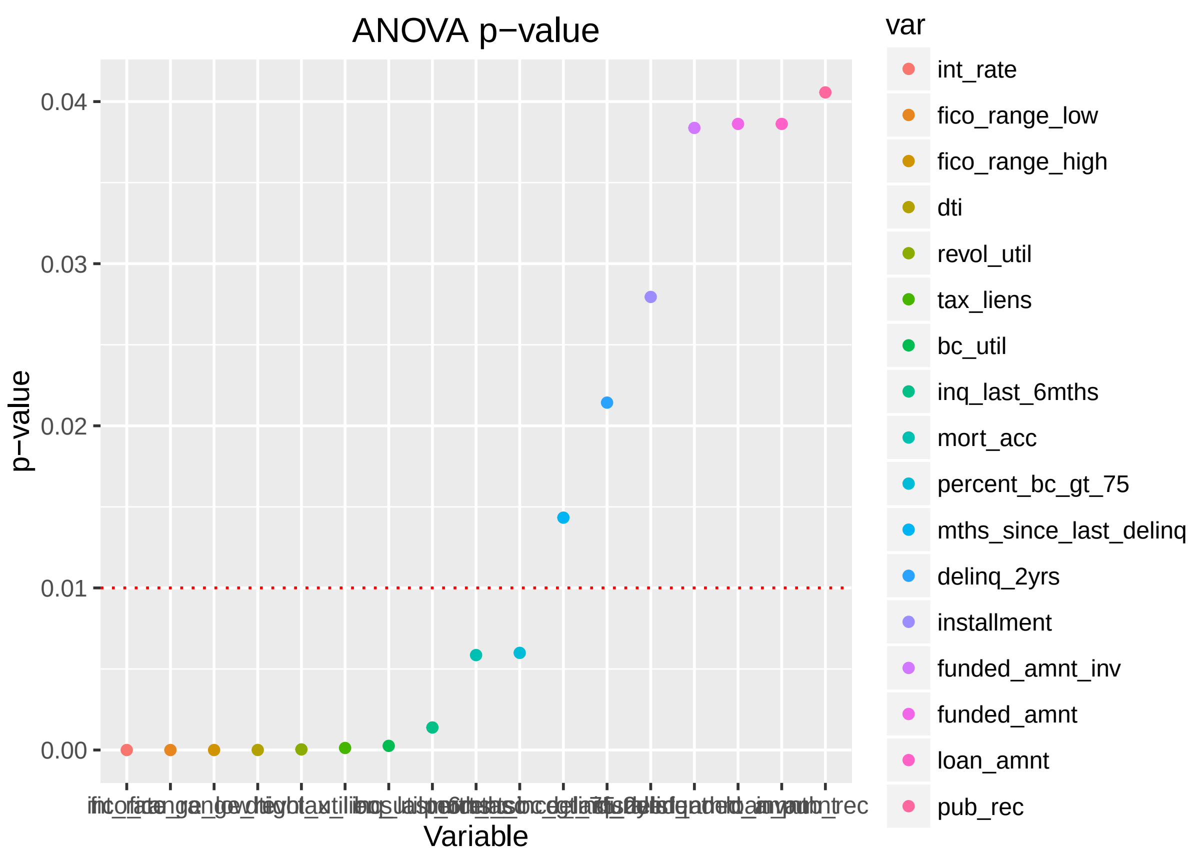
```
##   term emp_length home_ownership verification_status loan_status purpose
## 1    2         12              3                   3           7       1
##   addr_state
## 1         48
```

```
hasNA_all(lc)
```

```
##      mths_since_last_delinq      mths_since_last_record
##                        2421                        4302
##                  revol_util mths_since_last_major_derog
##                           3                        3674
##                     bc_util        mths_since_recent_bc
##                          77                          68
##    mths_since_recent_bc_dlq       mths_since_recent_inq
##                        3932                         381
##            num_tl_120dpd_2m            percent_bc_gt_75
##                         275                          77
```

ANOVA

```
lc1 <- lc[, !(colnames(lc) %in% c(names(n.factor_all(lc))))]
lc1$loan_status <- lc$loan_status
lc1 <- lc1 %>% subset(!loan_status == "Current")
anova.p <- data.frame(var = names(anova(lc1)), p.value = anova(lc1))
anova.p_plot <- anova.p %>% subset(p.value < 0.05) %>%
  mutate(var = reorder(x = var, X = p.value, min)) %>%
  ggplot(aes(x = var, y = p.value, color = var)) +
  geom_point() +
  geom_hline(yintercept = 0.01, colour = "red", linetype = 3) +
      labs(x = "Variable", y = "p-value", title = "ANOVA p-value")
anova.p_plot
```



```
sig.names <- as.character((anova.p %>% subset(p.value < 0.01))$var)
var.names <- c(sig.names, names(n.factor_all(lc)))
```

subset the data

```
lc2 <- lc[,c(var.names)]
lc2 <- lc2 %>% select(-purpose)
lc.categorical <- lc2[,names(n.factor_all(lc2))]
lc.categorical <- lc.categorical %>% select(-loan_status)
lc.numeric <- lc2[,!names(lc2) %in% names(n.factor_all(lc2))]
# Transform the categorical column to multiple numeric columns
categorical.list <- apply(lc.categorical, 2, function(x) model.matrix(~ x + 0))
```

Interested in the classification between Fully Paid and Potential Delinquency Events

```
lc3 <- as.data.frame(cbind(categorical.list, lc.numeric))
lc3$loan_status <- lc$loan_status
# Group the loan_status into 3 groups
lc3 <- lc3 %>% mutate(y = !(loan_status == "Current"))
lc3$y[lc3$loan_status == "Fully_Paid"] <- 2
```

```
lc3 <- lc3 %>% select(-loan_status)

## Logistic Regression
lc4 <- lc3 %>% subset(y != 0)
lc4$y[lc4$y == 2] <- 0 # Fully Paid set to be zero, while potential delinquency events set to be 1
log.f <- logistic(lc4)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred <- predict(log.f, lc4, type = "response")
```

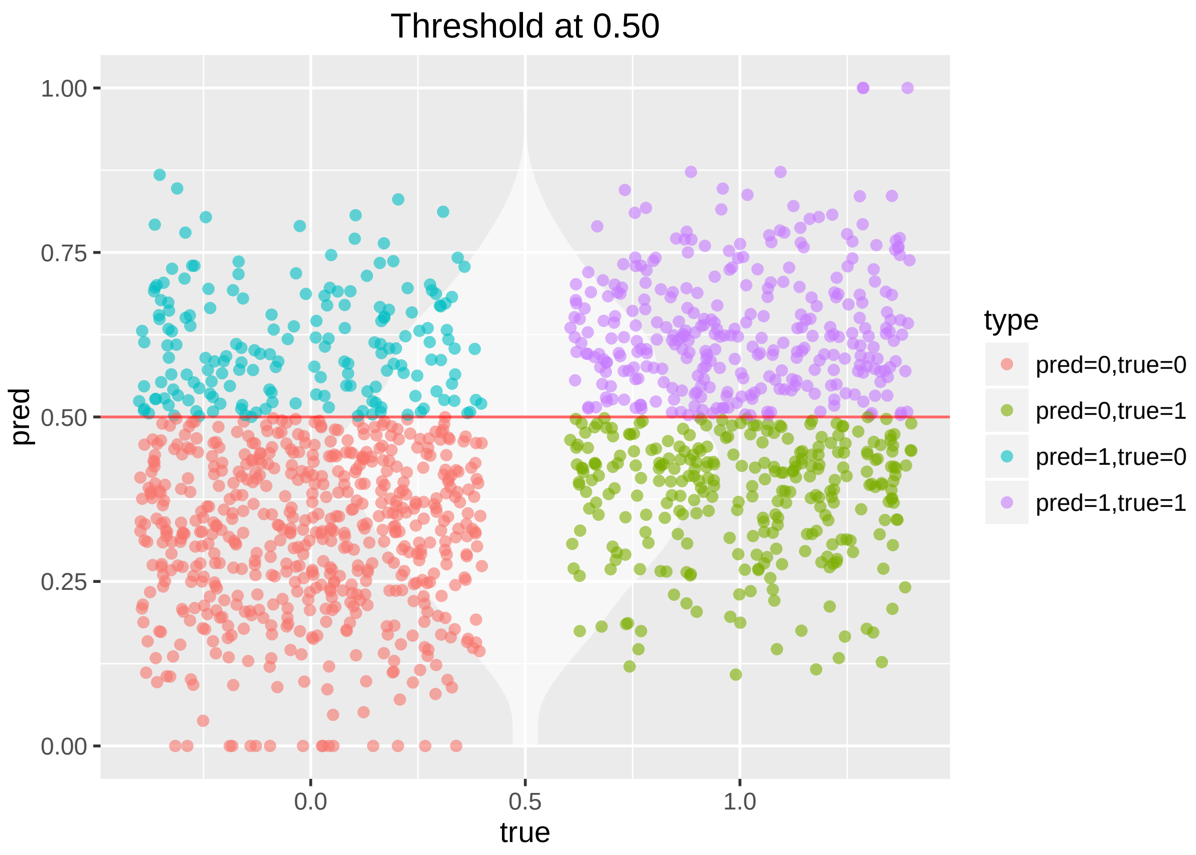
```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
# Compare the pred and true value
compare.data <- data.frame(pred = pred, true = lc4$y)

# in-sample prediction accuracy
# prediction accuracy when threshold = 0.5
pred1 = as.numeric(pred > 0.25)
1 - sum(pred1 == lc4$y, na.rm = T)/length(pred1)
```

```
## [1] 0.459695
```

```
# Prediction and true status when threshold = 0.5 plot
pred_type_plot <- plot_pred_type_distribution(compare.data, 0.5)
print(pred_type_plot)
```

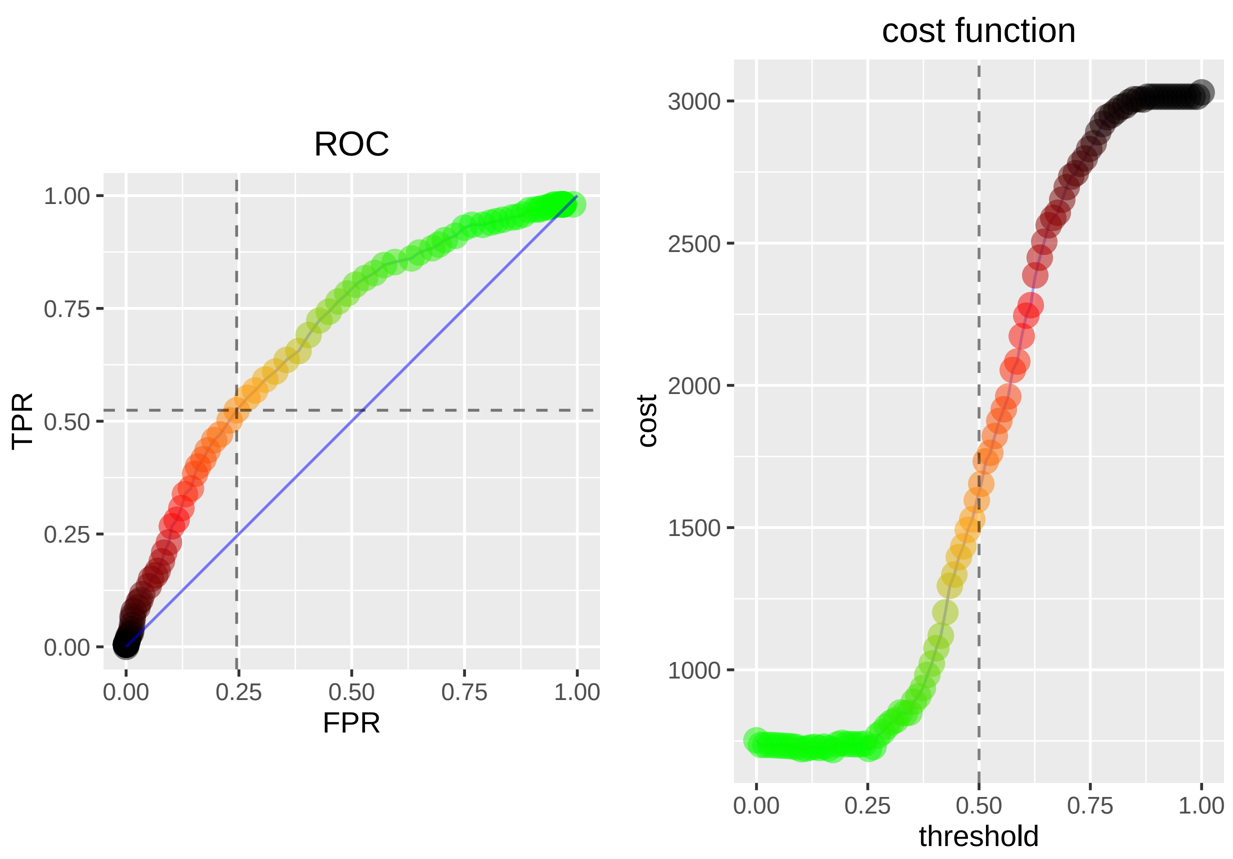
```
## Warning: Removed 19 rows containing non-finite values (stat_ydensity).

## Warning: Removed 19 rows containing missing values (geom_point).
```



```r
ggsave(filename = "FD_pred_type_plot.png", plot = pred_type_plot, path = ".",
       width = 10, height = 6, dpi = 400)
```

```
## Warning: Removed 19 rows containing non-finite values (stat_ydensity).

## Warning: Removed 19 rows containing missing values (geom_point).
```

```r
# roc, cost of FP = 1, cost of FN = 5
# Calculate the roc(FP: pred=1,true=0; FN:pred=0, true=1)
roc <- calculate_roc(compare.data, 1, 5, n = 100)
# Plot the roc, cost of FP = 1, cost of FN = 5
plot.roc <- plot_roc(roc, 0.5, 1, 5)
```



```r
print(plot.roc)
```

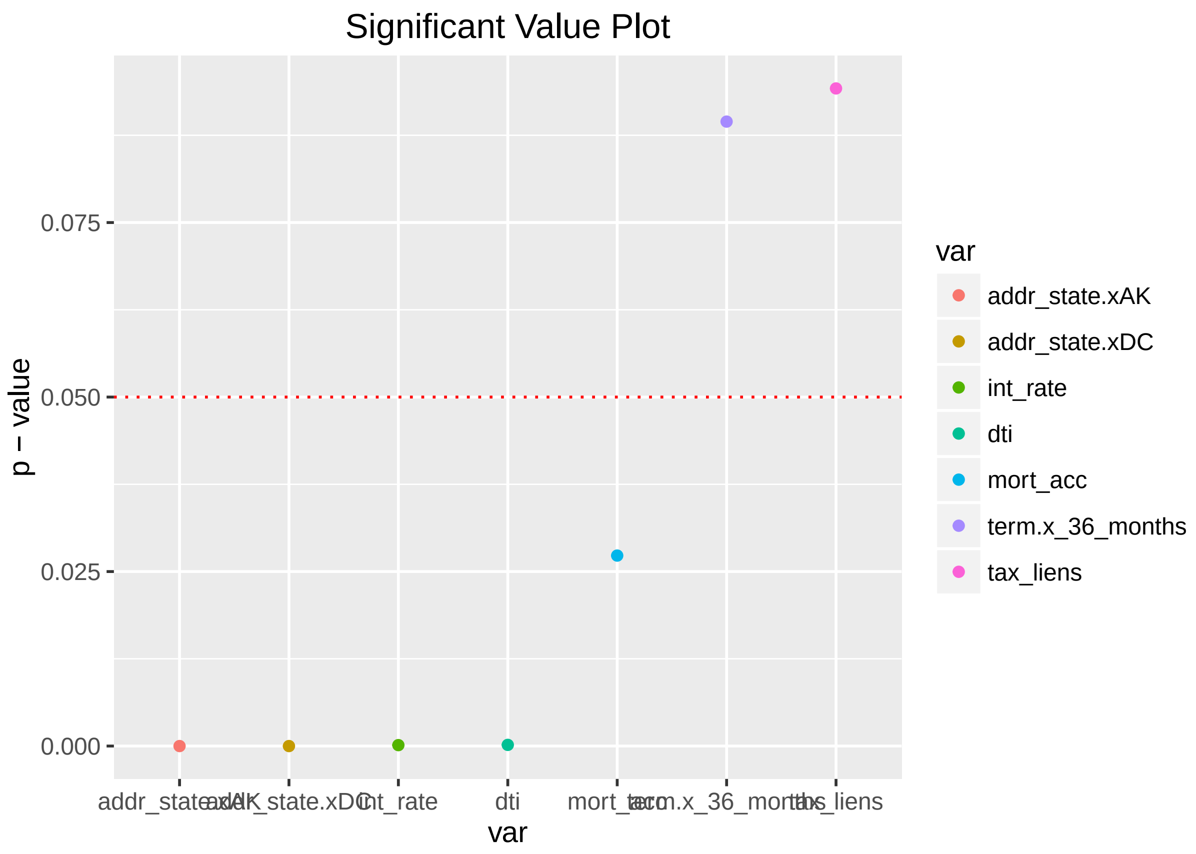
```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```r
ggsave(filename = "FD_plot_roc.png", plot = plot.roc, path = ".",
       width = 10, height = 6, dpi = 400)

# out-of-sample prediction accracy
cv_error <- cv_k(lc4)
```

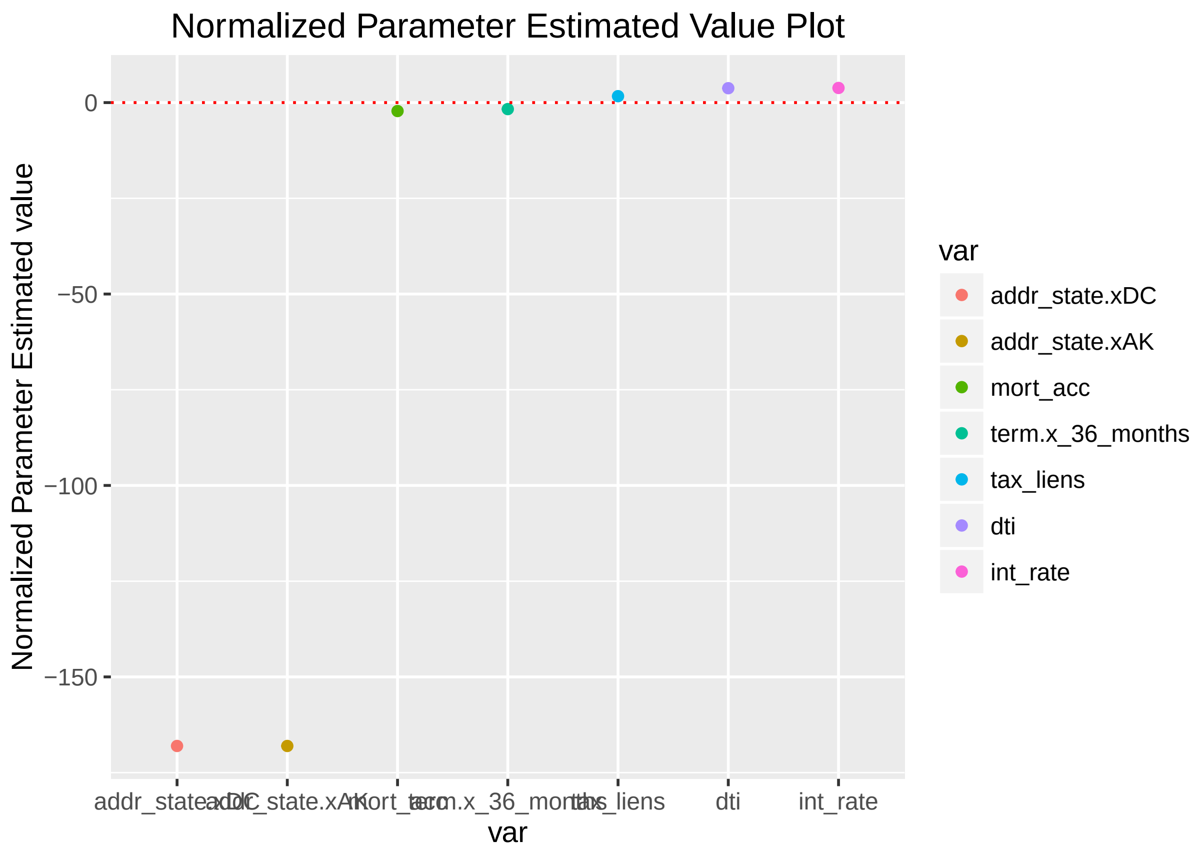
```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

```
## [1] 0.4133052
```

```
#Plot the significant value
significant_plot <- coeff_plot1(log.f, sig = T, 0.1)
print(significant_plot)
```



```
#Plot the coefficient value
coefficient_plot <- coeff_plot1(log.f, sig = F, 0.1)
print(coefficient_plot)
```



# Interested in the classification between Current and Potential Delinquency Events

```
lc3 <- as.data.frame(cbind(categorical.list, lc.numeric))
lc3$loan_status <- lc$loan_status
# Group the loan_status into 3 groups
lc3 <- lc3 %>% mutate(y = !(loan_status == "Fully_Paid"))
lc3$y[lc3$loan_status == "Current"] <- 2
lc3 <- lc3 %>% select(-loan_status)

## Logistic Regression
lc4 <- lc3 %>% subset(y != 0)
lc4$y[lc4$y == 2] <- 0 # Fully Paid set to be zero, while potential delinquency events set to be 1
log.f <- logistic(lc4)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

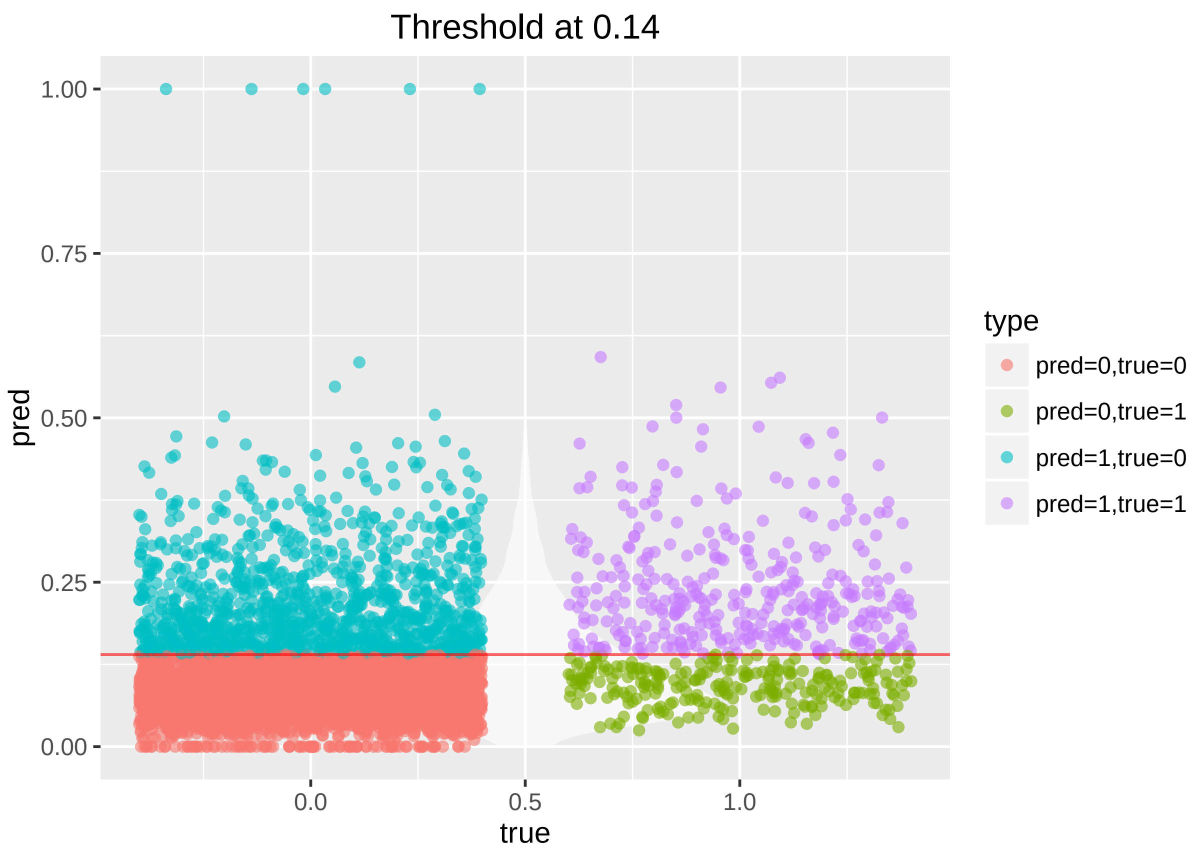
```
pred <- predict(log.f, lc4, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
# Compare the pred and true value
compare.data <- data.frame(pred = pred, true = lc4$y)

# in-sample prediction accuracy
# prediction accuracy when threshold = 0.5
pred1 = as.numeric(pred > 0.14)
1 - sum(pred1 == lc4$y, na.rm = T)/length(pred1)
```

```
## [1] 0.3058173
```

```
# Prediction and true status when threshold = 0.5 plot
pred_type_plot <- plot_pred_type_distribution(compare.data, 0.14)
print(pred_type_plot)
```

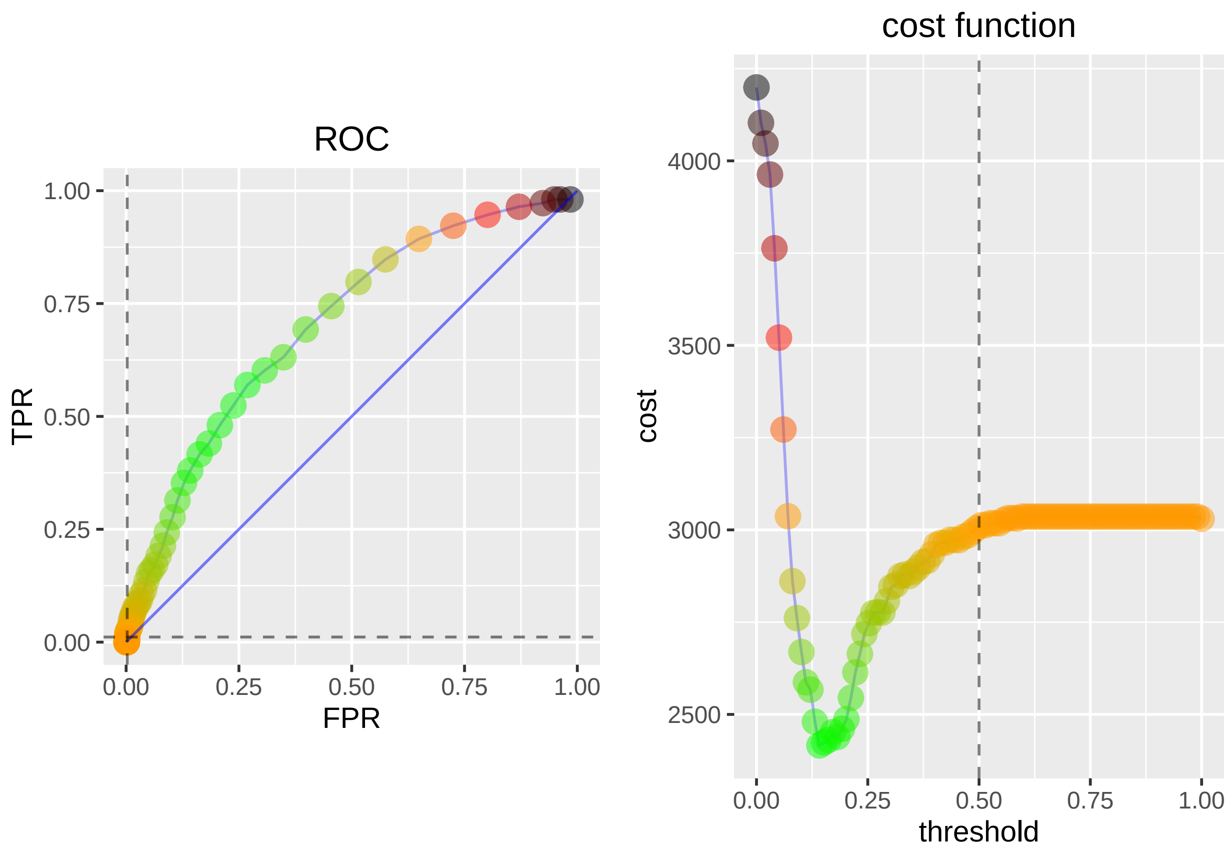
```
## Warning: Removed 77 rows containing non-finite values (stat_ydensity).

## Warning: Removed 77 rows containing missing values (geom_point).
```



```
ggsave(filename = "CD_pred_type_plot.png", plot = pred_type_plot, path = ".",
       width = 10, height = 6, dpi = 400)
```

```
## Warning: Removed 77 rows containing non-finite values (stat_ydensity).

## Warning: Removed 77 rows containing missing values (geom_point).
```

```
# roc, cost of FP = 1, cost of FN = 5
# Calculate the roc(FP: pred=1,true=0; FN:pred=0, true=1)
roc <- calculate_roc(compare.data, 1, 5, n = 100)
# Plot the roc, cost of FP = 1, cost of FN = 5
plot.roc <- plot_roc(roc, 0.5, 1, 5)
```



```
print(plot.roc)
```

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

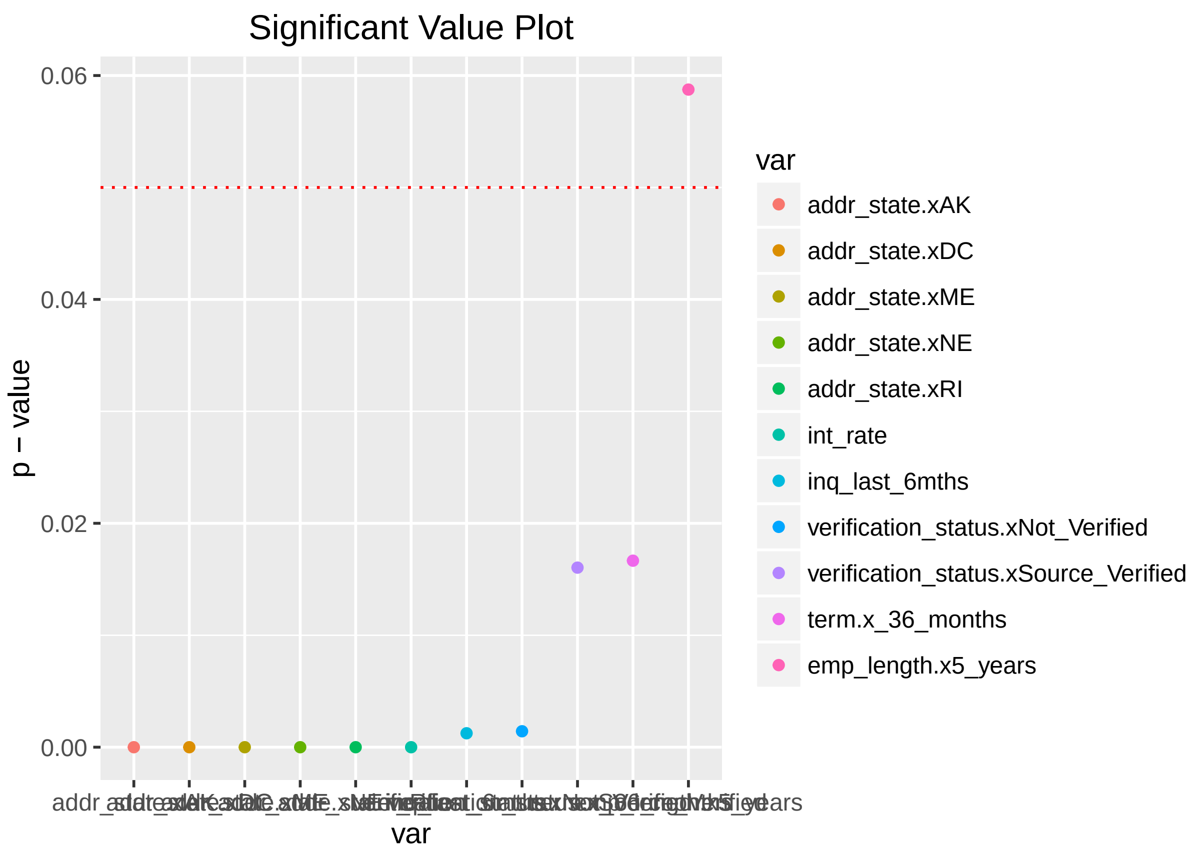
```
ggsave(filename = "CD_plot_roc.png", plot = plot.roc, path = ".",
       width = 10, height = 6, dpi = 400)

# out-of-sample prediction accracy
cv_error <- cv_k(lc4, threshold = 0.14)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

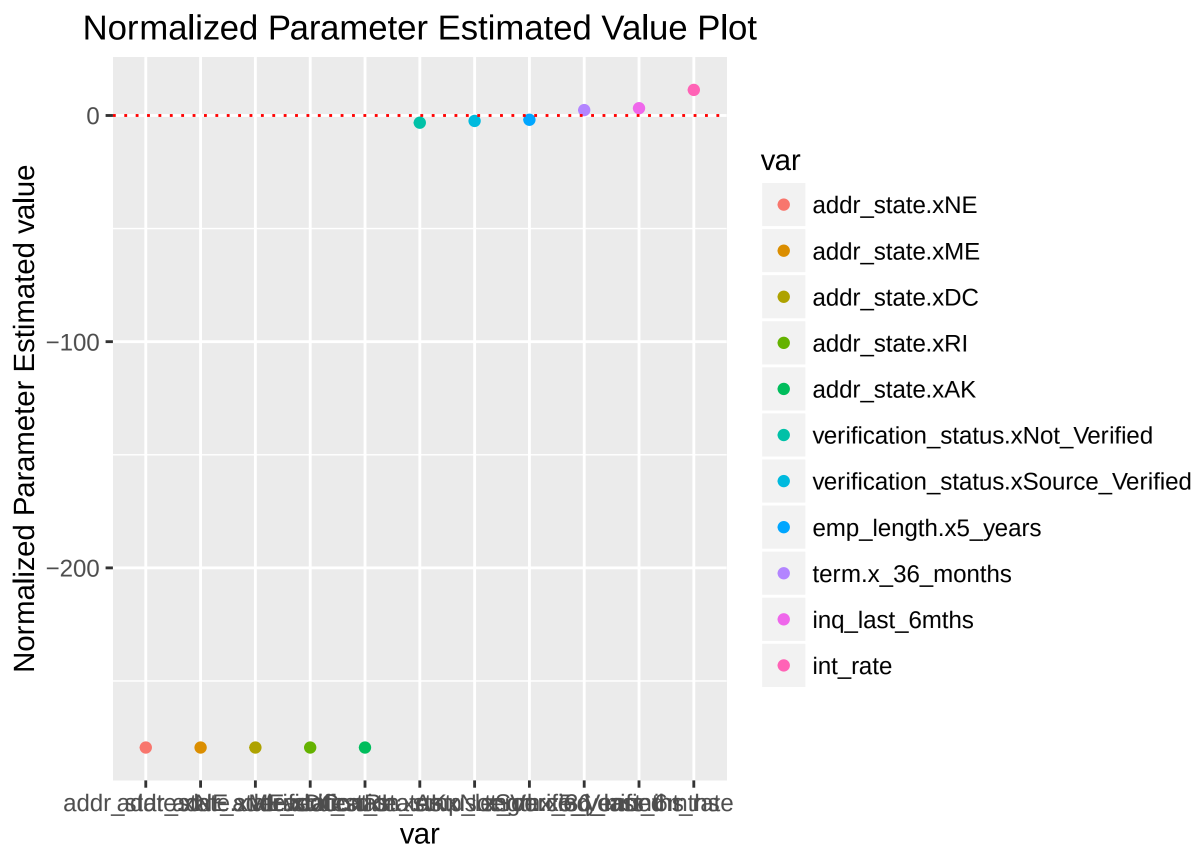
```
cv_error
```

```
## [1] 0.3111318
```

```
#Plot the significant value
significant_plot <- coeff_plot1(log.f, sig = T, 0.1)
print(significant_plot)
```



```
#Plot the coefficient value
coefficient_plot <- coeff_plot1(log.f, sig = F, 0.1)
print(coefficient_plot)
```



# Interested in classification between Current and Fully Paid

```
lc3 <- as.data.frame(cbind(categorical.list, lc.numeric))
lc3$loan_status <- lc$loan_status
# Group the loan_status into 3 groups
lc3 <- lc3 %>% mutate(y = (loan_status == "Fully_Paid"))
lc3$y[lc3$loan_status == "Current"] <- 2
lc3 <- lc3 %>% select(-loan_status)

## Logistic Regression
lc4 <- lc3 %>% subset(y != 0)
lc4$y[lc4$y == 2] <- 0 # Fully Paid set to be zero, while potential delinquency events set to be 1
log.f <- logistic(lc4)
pred <- predict(log.f, lc4, type = "response")
```

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

```
# Compare the pred and true value
compare.data <- data.frame(pred = pred, true = lc4$y)
```

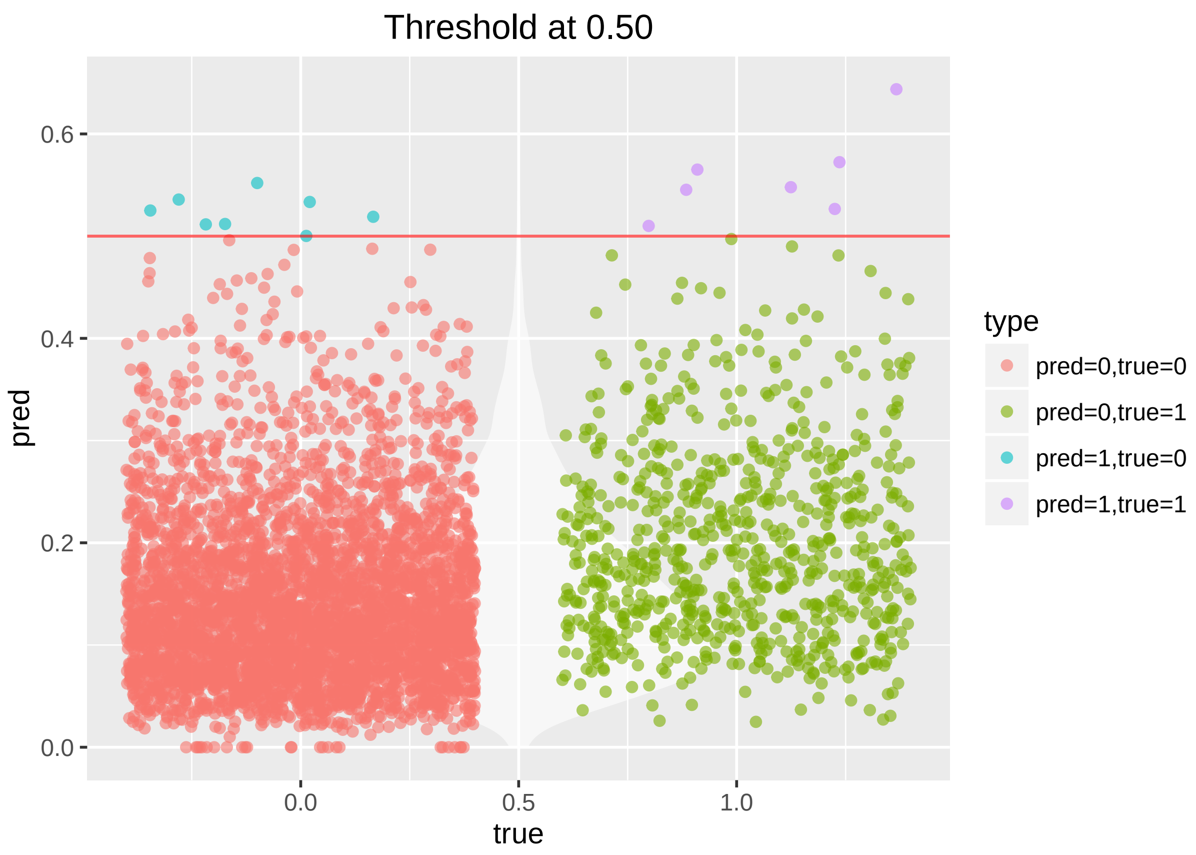
```r
# in-sample prediction accuracy
# prediction accuracy when threshold = 0.5
pred1 = as.numeric(pred > 0.5)
1 - sum(pred1 == lc4$y, na.rm = T)/length(pred1)
```

```
## [1] 0.1642445
```

```r
# Prediction and true status when threshold = 0.5 plot
pred_type_plot <- plot_pred_type_distribution(compare.data, 0.5)
print(pred_type_plot)
```

```
## Warning: Removed 72 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 72 rows containing missing values (geom_point).
```

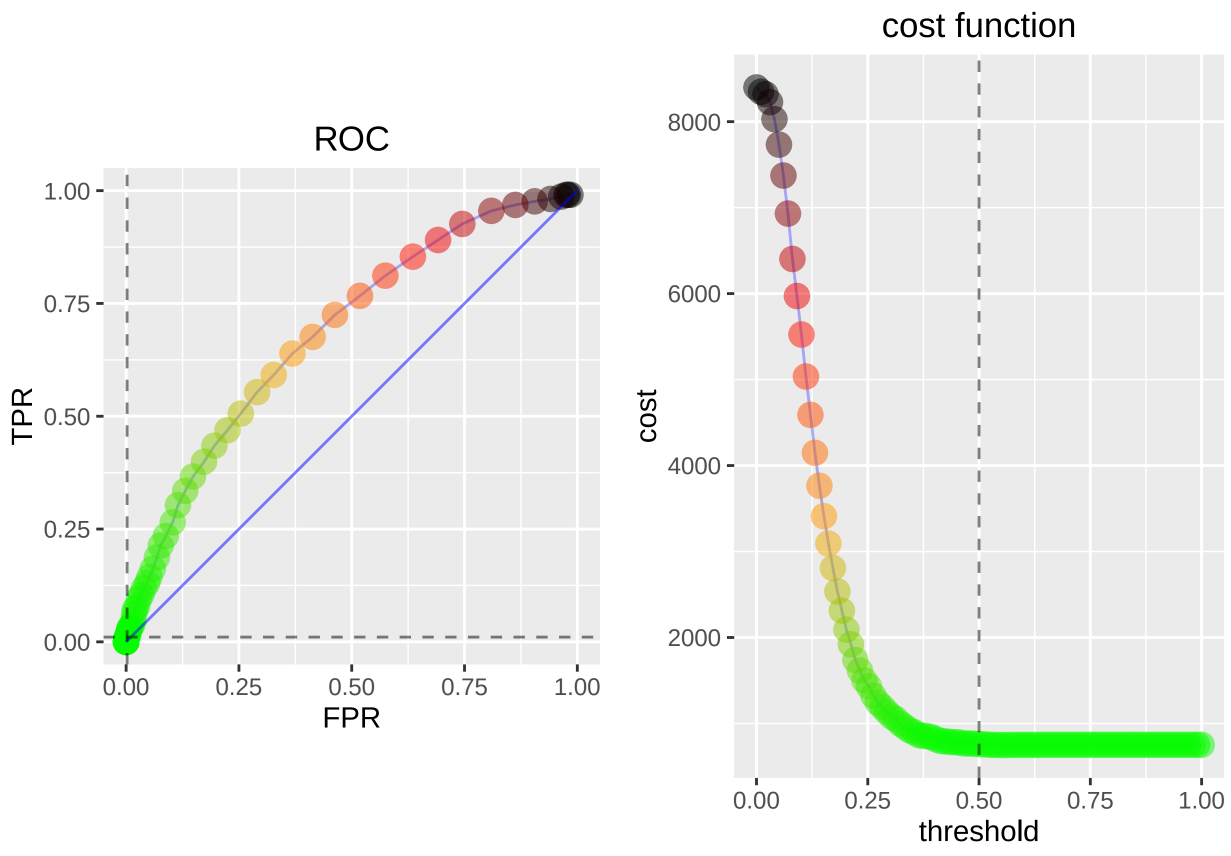


```r
ggsave(filename = "FC_pred_type_plot.png", plot = pred_type_plot, path = ".",
       width = 10, height = 6, dpi = 400)
```

```
## Warning: Removed 72 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 72 rows containing missing values (geom_point).
```

```r
# roc, cost of FP = 1, cost of FN = 5
# Calculate the roc(FP: pred=1,true=0; FN:pred=0, true=1)
roc <- calculate_roc(compare.data, 2, 1, n = 100)
# Plot the roc, cost of FP = 1, cost of FN = 5
plot.roc <- plot_roc(roc, 0.5, 2, 1)
```



```r
print(plot.roc)
```

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```r
ggsave(filename = "FC_plot_roc.png", plot = plot.roc, path = ".",
       width = 10, height = 6, dpi = 400)

# out-of-sample prediction accracy
cv_error <- cv_k(lc4)
```

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

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

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

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

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

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

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

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

```
## Warning: glm.fit: algorithm did not converge
```

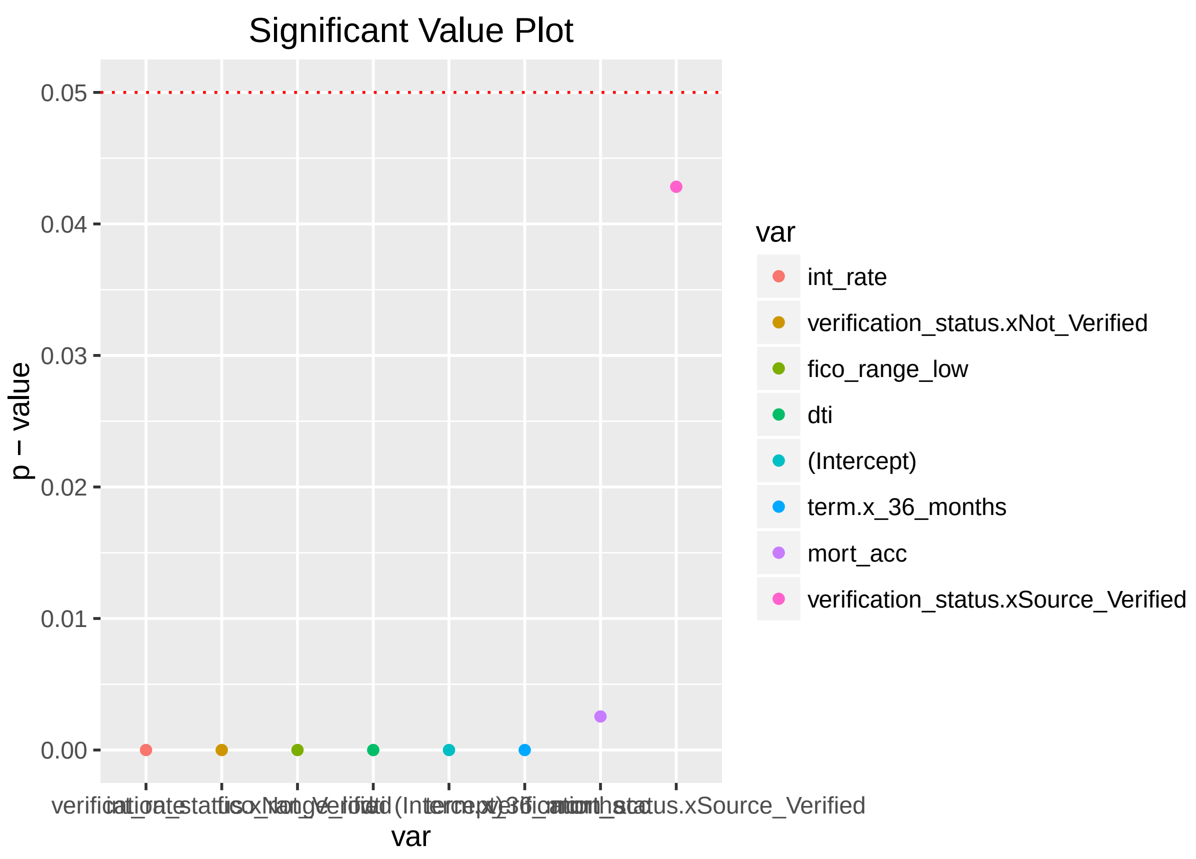
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

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

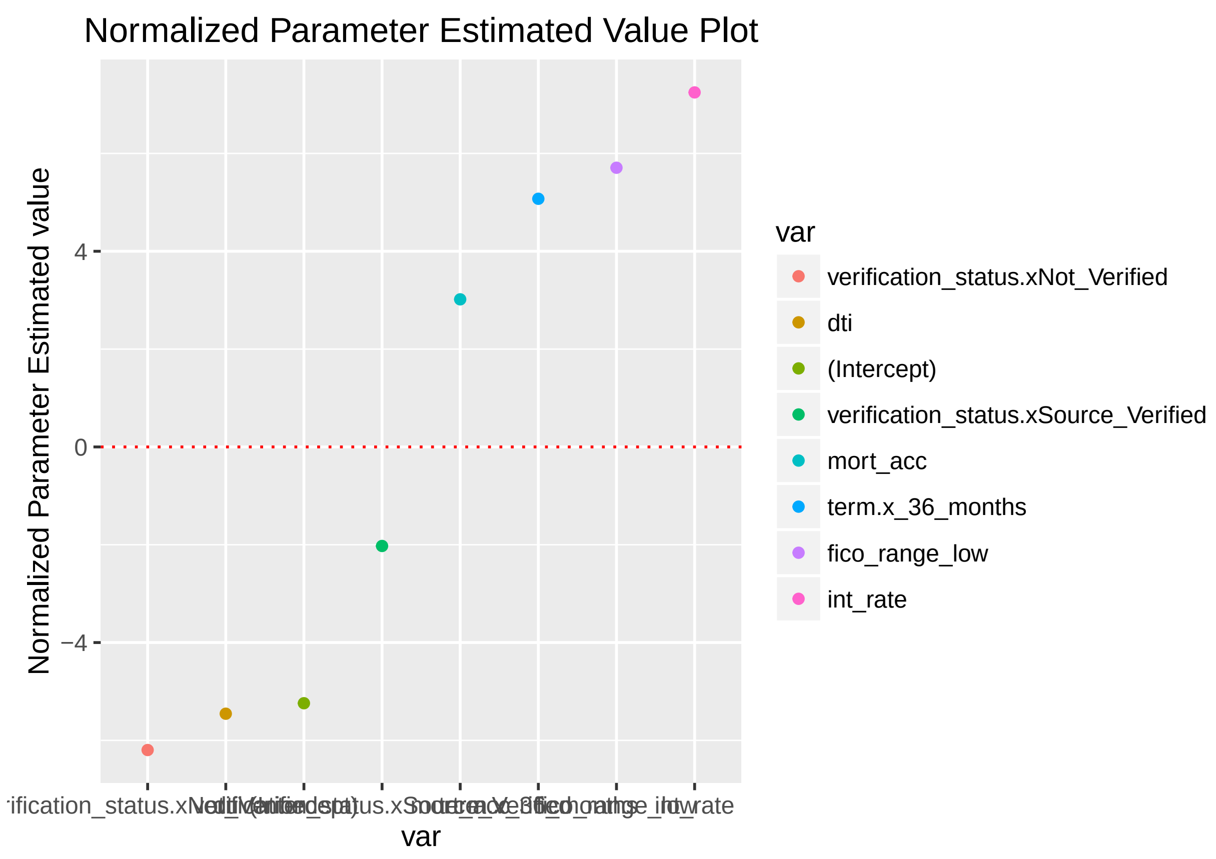
```r
cv_error
```

```
## [1] 0.1650563
```

```r
#Plot the significant value
significant_plot <- coeff_plot1(log.f, sig = T, 0.1)
print(significant_plot)
```



```r
#Plot the coefficient value
coefficient_plot <- coeff_plot1(log.f, sig = F, 0.1)
print(coefficient_plot)
```



```r
# What is the prediction for Status "Current"?
current <- lc3 %>% subset(y == 0)
pred.current <- predict(log.f, current, type = "response")
```

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

Kernel SVM

```r
# Prepare for kernel SVM
lc4 <- lc4[-which(is.na(lc4$percent_bc_gt_75) | is.na(lc4$revol_util) | is.na(lc4$bc_util)),]

l.kern <- svm(lc4, "rbfdot")
l.kern
# New data
newdata <- as.matrix(lc4%>% select(-y))
pred.kern <- predict(l.kern, newdata)
###
compare.data <- data.frame(pred = pred.kern, true = lc4$y)

# in-sample prediction accuracy
# prediction accuracy when threshold = 0.5
pred1 = as.numeric(pred > 0.14)
```

```r
1 - sum(pred1 == lc4$y, na.rm = T)/length(pred1)
# Prediction and true status when threshold = 0.5 plot
plot_pred_type_distribution(compare.data, 0.5)
# roc, cost of FP = 1, cost of FN = 5
# Calculate the roc(FP: pred=1,true=0; FN:pred=0, true=1)
roc <- calculate_roc(compare.data, 1, 5, n = 100)
# Plot the roc, cost of FP = 1, cost of FN = 5
plot.roc <- plot_roc(roc, 0.5, 1, 5)

# in-sample prediction accuracy
1 - sum(pred.kern == lc4$y, na.rm = T)/length(lc4$y)
# cross validation error
cv.kernel.error <- cv_ksvm(lc4, "vanilladot", k = 10)
cv.kernel.error
```