EDA_Modeling_Data Wrangling

Modeling Methods: ***Linear Regression, Lasso (L1 penalty) and Elastic Net Regression***

Prepared by: Kim H Nguyen, Data Analyst Candidate

E-mail: hn13@brandeis.edu / kim.ng0913@gmail.com

```
## Call Libraries:
library(sqldf)

library(dplyr)

library(tidyverse)

library(tidyr)
library(ggplot2)
library(Hmisc)

library(taRifx)

library(psych)

library(caret)

library(glmnet)

library(mlbench)

## Read in Data Sets
user_identifiers <- read.csv('C:/Users/hn13/Desktop/loan_data/user_identifier
s.csv')
loan_data <- read.csv('C:/Users/hn13/Desktop/loan_data/loan_data.csv')
```

Part I - Data Wrangling of 'user_identifiers'

```
# Extract Numeric Values (yyyymmdd) from Strings (Name) and create Date of Bi
rth (DoB) column
user_dob <- user_identifiers %>%
  mutate(DoB = destring(UserId, keep="0-9.-"))

# Extract Year of Birth from Numeric Value and create the Year of Birth (YOB)
column
user_yob <- user_dob %>%
  mutate(Year = substring(DoB,1,4))

# Create a new column "Current_Year" = 2018
```

```r
current_year <- user_yob %>%
  mutate(Current_Year = '2018')

# Calculate users' age and order "Age" in ascending order
user_age <- current_year %>%
            mutate(Age = as.numeric(Current_Year) - as.numeric(Year)) %>%
            arrange(Age)

# Statistical summary and distribution of users' age
describe(user_age$Age)

##    vars   n  mean     sd median trimmed   mad min  max range  skew
## X1    1 299 49.66 127.15     39   40.47 16.31  -2 2217  2219 16.57
##    kurtosis   se
## X1   279.36 7.35

#Detect records with unsual values for "Age" by ascending order
head(user_age)

##                             UserId      DoB Year Current_Year Age
## 1 DanielleAtkinsonHerring20201209 20201209 2020         2018  -2
## 2             MarthaMilroy20170601 20170601 2017         2018   1
## 3          JahkeyiaSanchez19971117 19971117 1997         2018  21
## 4               JasonMurry19970810 19970810 1997         2018  21
## 5          ThomasGunderson19970711 19970711 1997         2018  21
## 6           EmilySouthern19970107 19970107 1997         2018  21

#Detect records with outliers, irregular/unusual values for "Age":
age_outliers <- sqldf("SELECT UserId, Year, Age
                      FROM user_age
                      WHERE Age > 71")
age_outliers

##                        UserId Year  Age
## 1      JaniceRoberts19220115 1922   96
## 2      CristinaJones19181108 1918  100
## 3  CassandraThompson19180528 1918  100
## 4     LazarickSpeats19180602 1918  100
## 5     AlexanderLopez19180608 1918  100
## 6       RobertDentis19180608 1918  100
## 7       DavidGilarde19180612 1918  100
## 8     KevinVardanian19181012 1918  100
## 9        BrockLesnar18000101 1800  218
## 10  PeggySilva-Sword19900404 -199 2217

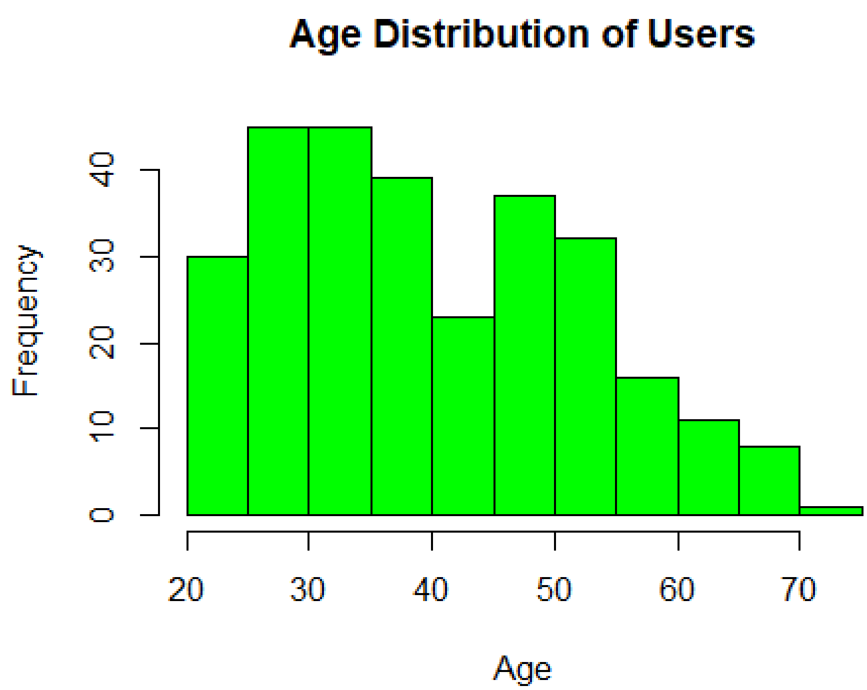
# Identify users whose age information is missing

age_missing <- sqldf("SELECT *
                      FROM user_age
                      WHERE Age IS NULL")
age_missing
```

```
##     UserId DoB Year Current_Year Age
## 1 AaronTan  NA <NA>         2018  NA

# Select a subset of users'age from the range between 20 and 70, exclude all outliers and typos:
age_20_71 <- sqldf("SELECT *
                   FROM user_age
                   WHERE AGE BETWEEN 20 AND 71")
#plot histogram of age distribution for 286 users:

hist(age_20_71$Age,
     col = 'green',
     xlab = 'Age',
     main = 'Age Distribution of Users')
```



Part II - Explorator Data Analysis of Loan Data:

```
#Question: What is the total loan amount and number of borrowers by States?

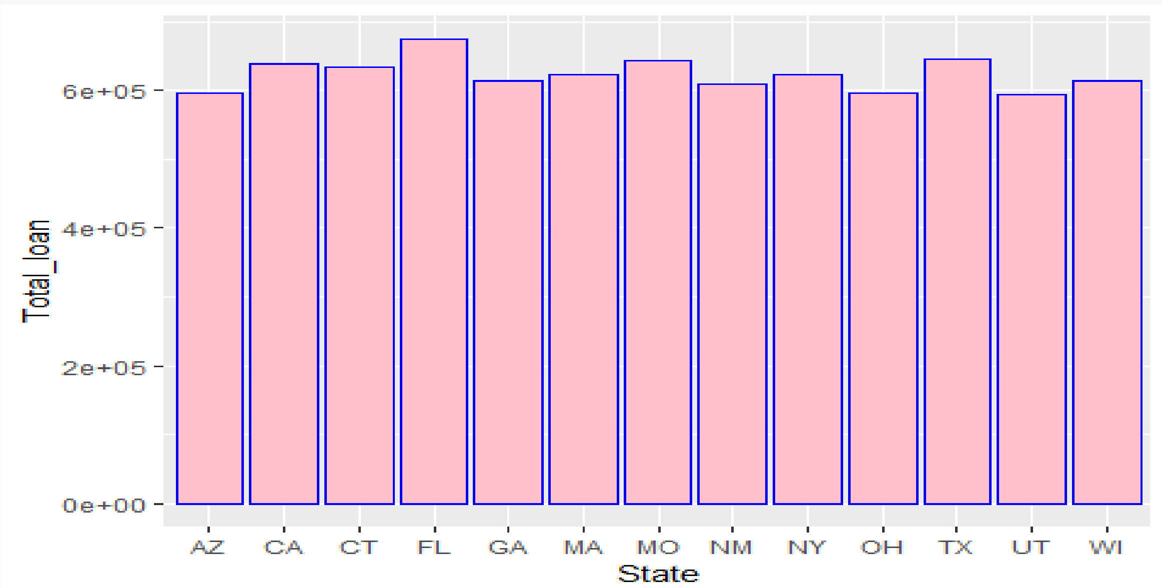
#Florida has the largest number of loan applicants with 827 borrowers. This makes it the State with highest loan amount of $674,775, which was followed by Texas, Missouri and California with a small difference of almost $30,000. Overall, the total loan amount is not significantly different across States.

amount_by_state <- loan_data %>%
         group_by(State) %>%
         summarise(Total_loan = sum(Amount),
```

```
                   Total_borrowers = n(),
                  Mean_loan = mean(Amount),
                   Sd_loan = sd(Amount),
                   Max_loan = max(Amount),
                   Min_loan = min(Amount)) %>%
         arrange(desc(Total_loan))
amount_by_state

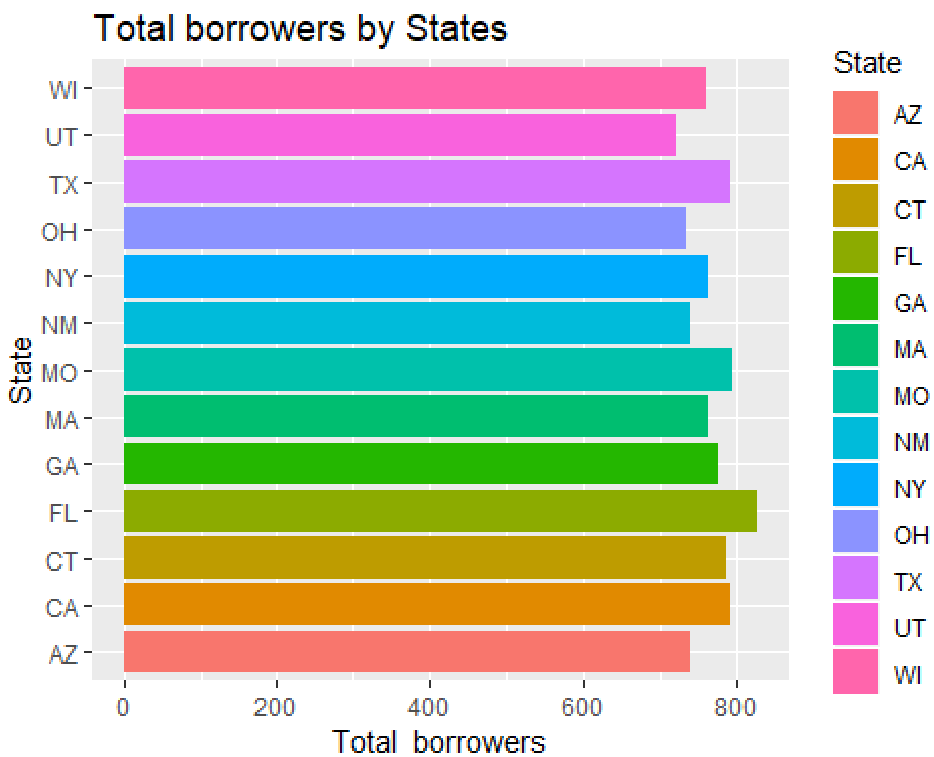
## # A tibble: 13 x 7
##    State Total_loan Total_borrowers Mean_loan Sd_loan Max_loan Min_loan
##    <fct>      <int>           <int>     <dbl>   <dbl>    <dbl>    <dbl>
##  1 FL        674775             827      816.    301.     1500      300
##  2 TX        645900             794      813.    299.     1500      300
##  3 MO        643300             795      809.    295.     1500      300
##  4 CA        638025             794      804.    293.     1500      300
##  5 CT        633475             787      805.    292.     1500      300
##  6 NY        623825             763      818.    297.     1500      300
##  7 MA        622725             765      814.    295.     1500      300
##  8 GA        614975             777      791.    283.     1500      300
##  9 WI        614875             761      808.    292.     1500      300
## 10 NM        608550             741      821.    308.     1500      300
## 11 AZ        595150             741      803.    291.     1500      300
## 12 OH        594925             734      811.    291.     1500      300
## 13 UT        592925             721      822.    287.     1500      300

# Barplot of Total_loan by States:
#In terms of the variability of loan amount, Florida also has the largest sta
ndard deviation. However, Utah has the largest mean amount.
p<-ggplot(data= amount_by_state, aes(x= State, y= Total_loan)) +
  geom_bar(stat="identity", color="blue", fill = 'pink')
p
```



```
#Barplot of Total number of Borrowers By States:

ggplot(amount_by_state, aes(x=State, y = Total_borrowers, fill = State)) +
         geom_col() +
         coord_flip() +
         ggtitle('Total borrowers by States')
```

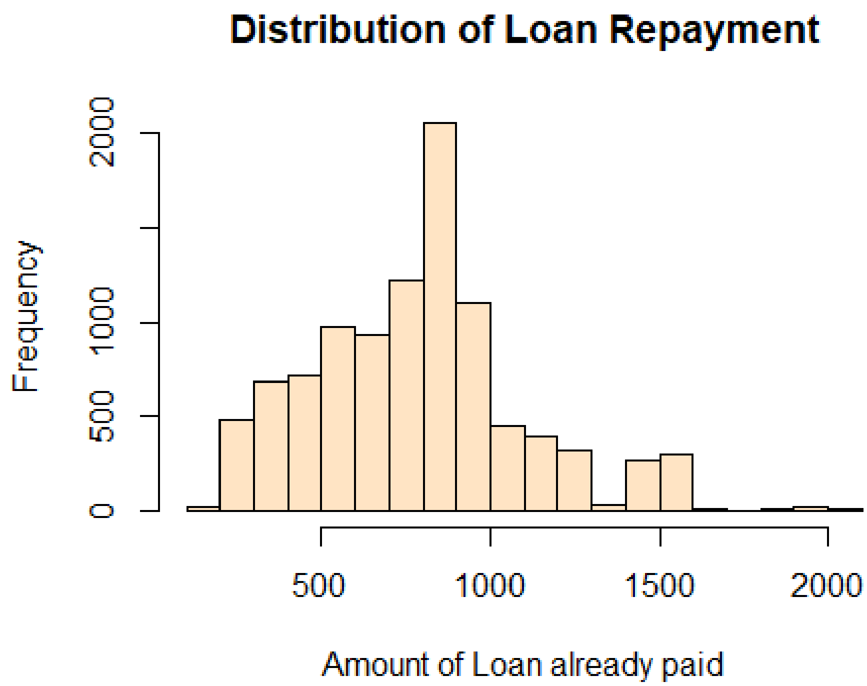



```
# What is statistical summary and overall distribution of Loan Repayment:
#The loan Repayment ranges from $186 to 2068 with the similar mean and median
value, which indicate an even distribution around the central tendency.
repayment_summary <- loan_data %>%
    summarise(Mean_Repayment = mean(Loan.Repayment),
            Median_Repayment = median(Loan.Repayment),
            Sd_Repayment = sd(Loan.Repayment),
            Max_Repayment = max(Loan.Repayment),
            Min_Repayment = min(Loan.Repayment))
repayment_summary

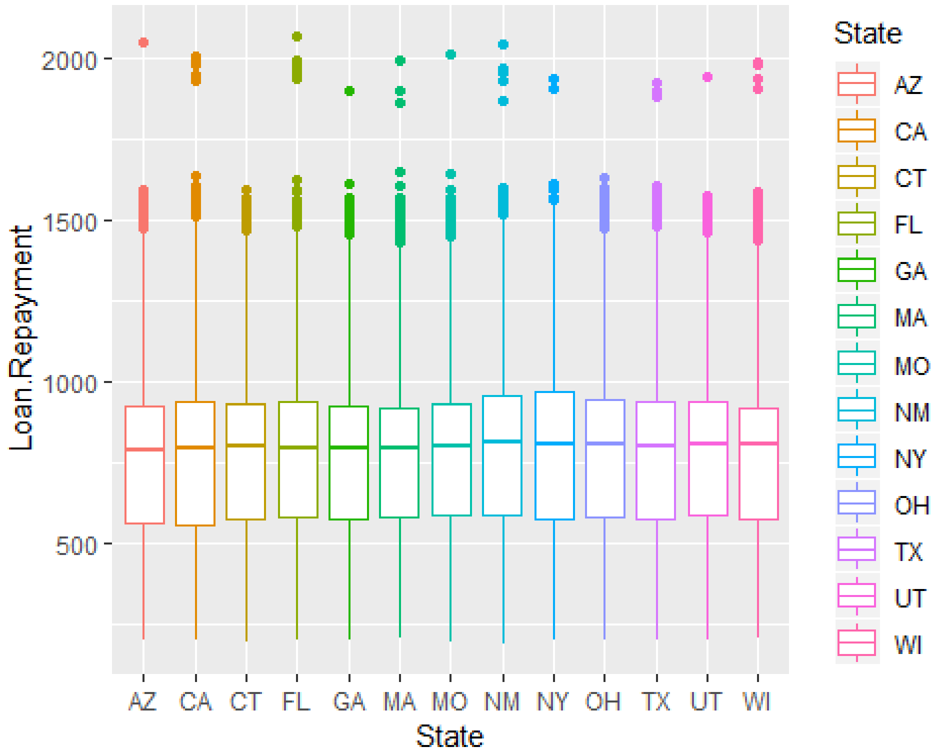
##   Mean_Repayment Median_Repayment Sd_Repayment Max_Repayment Min_Repayment
## 1       785.9376              798     310.2511          2068           186

# Overall distribution of Loan Repayment:
# 75% of total Loan Repayment is less than $1000 and 95% of total Loan Repaym
ent is less than $1,500.
# There are five extreme values of Loan Repayment that are greater than 2,000
.
hist(loan_data$Loan.Repayment,
```

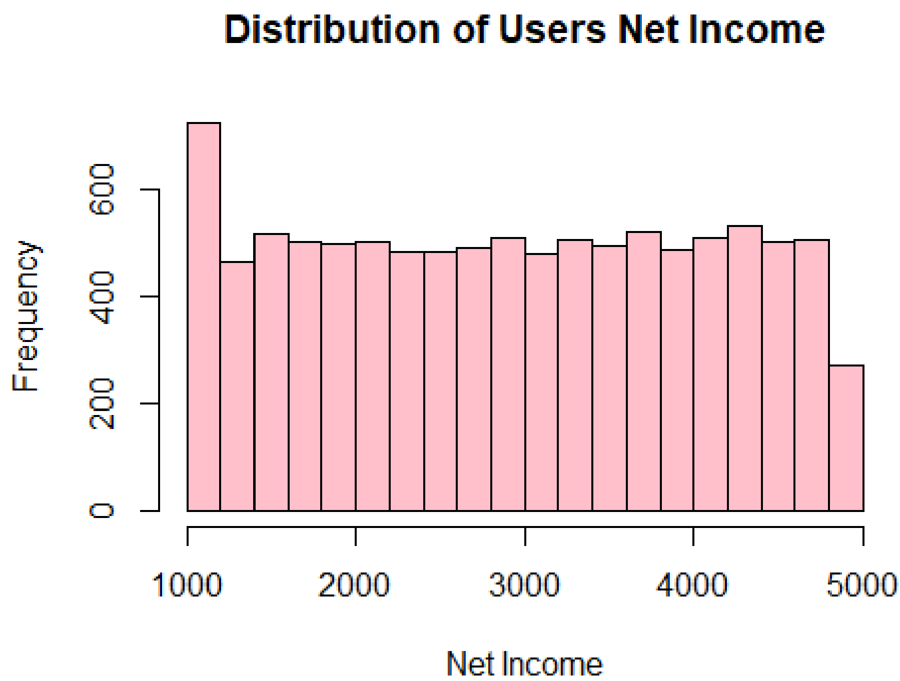
```
    col = 'bisque',
    xlab = 'Amount of Loan already paid',
    main = 'Distribution of Loan Repayment')
```




```
# Box Plot of Loan Repayment by States
# At State-level, the distribution of Loan Repayment is almost the same. FL,
CA, NM and MA are the States with more extreme value for loan amount.
loan_data %>%
         group_by(State) %>%
         ggplot(aes(x= State, y= Loan.Repayment, col = State)) +
         geom_boxplot()
```



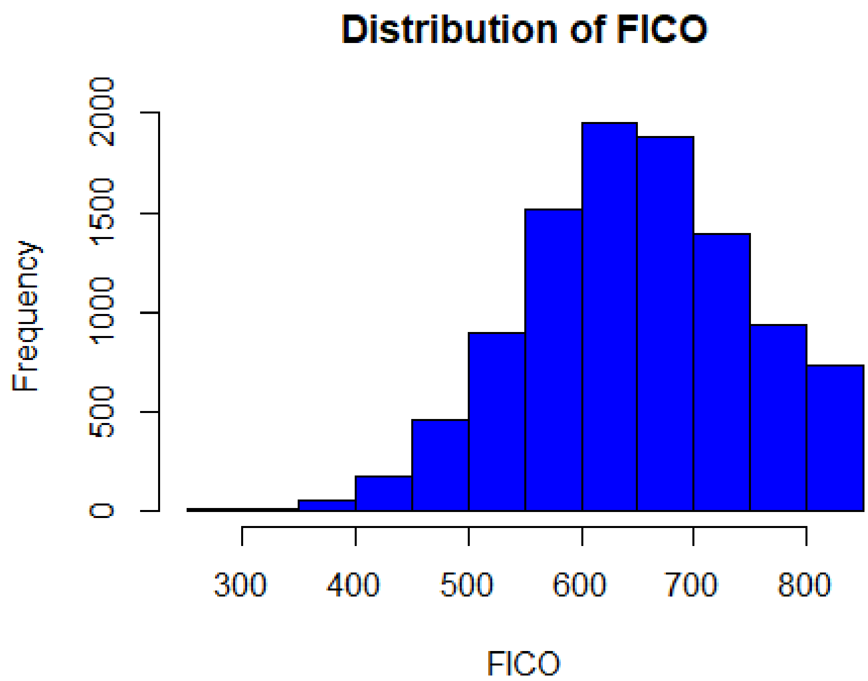

```
#Question: What is the distribution of Net Income?
#The Net Income evenly distribute from $1,000 to $5,000 with the similar mean
and median of almost $3000.
#25 percent of Net Income is below $2,000 and about 10% of Net Income lies in
the highest range from 4,000 to 5,000.
hist(loan_data$Net.Income,
     col = 'pink',
     xlab = 'Net Income',
     main = 'Distribution of Users Net Income')
```




```
#Question: What is the median and mean of Net Income?
summary(loan_data$Net.Income)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1000    2000    3000    2969    4000    4900

#Distribution of FICO across States:
#The most frequent FICO scores lie in the range from 500-750, around the over
all mean of around 650. At the state-level, there is not much difference in t
he average FICO score, which centers around 650 for each State.
hist(loan_data$FICO,
     col = 'blue',
     xlab = 'FICO',
     main = 'Distribution of FICO')
```



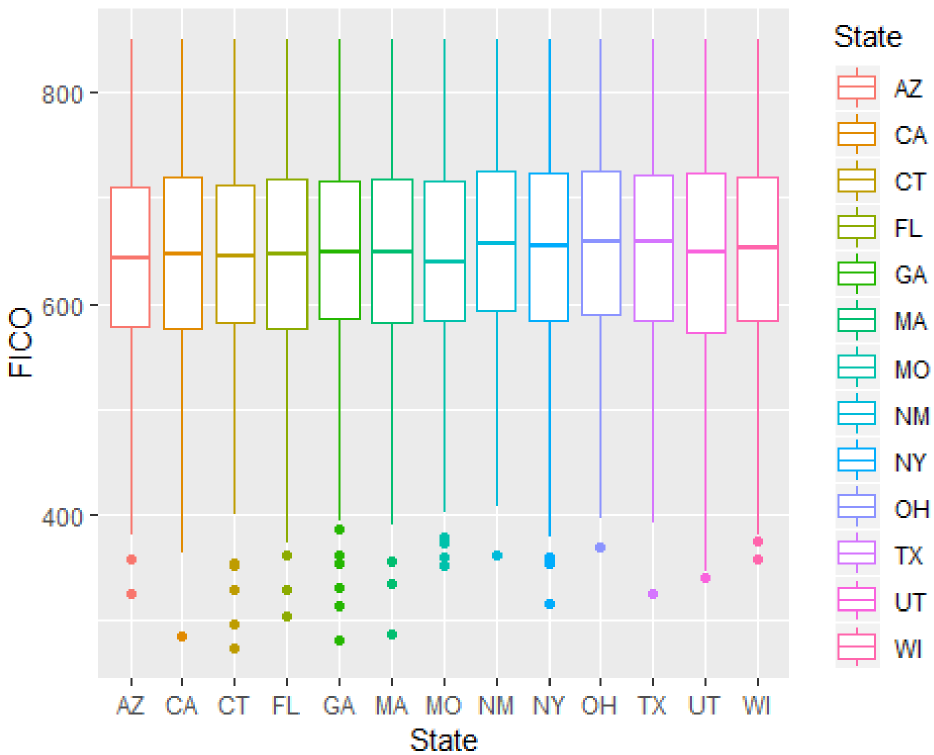

```
# Statistical summary of FICO by State:
#There are a wide variability of FICO score in UT, NY, CA and FL as they have
large standard deviation.
fico_by_state <- loan_data %>%
         group_by(State) %>%
         summarise(Sd_fico = sd(FICO),
                   Min_fico = min(FICO),
                   Max_fico = max(FICO),
                   Mean_fico = mean(FICO),
                   Median_fico = median(FICO)) %>%
         arrange(desc(Sd_fico))

fico_by_state

## # A tibble: 13 x 6
##    State Sd_fico Min_fico Max_fico Mean_fico Median_fico
##    <fct>   <dbl>    <dbl>    <dbl>     <dbl>       <dbl>
##  1 UT      105.       340      850      647.         648
##  2 NY      102.       316      850      653.         654
##  3 CA      101.       285      850      647.         646
##  4 FL      101.       305      850      647.         647
##  5 CT       99.8      274      850      646.         645
##  6 GA       99.0      281      850      648.         648
##  7 MA       98.5      287      850      649.         649
##  8 WI       98.4      358      850      650.         653
##  9 MO       98.0      353      850      646.         639
## 10 AZ       97.9      326      850      643.         643
```

```
## 11 NM        97.2       362       850       657.          656
## 12 OH        96.9       370       850       659.          659
## 13 TX        96.1       326       850       653.          658
```

```
# Box Plot of FICO by States:
# States that have several loan applicants whose FICO scores are extremely lo
w under 300 are CT (two outliers) GA, MA, CA.
loan_data %>%
         group_by(State) %>%
         ggplot(aes(x= State, y= FICO, col = State)) +
         geom_boxplot()
```

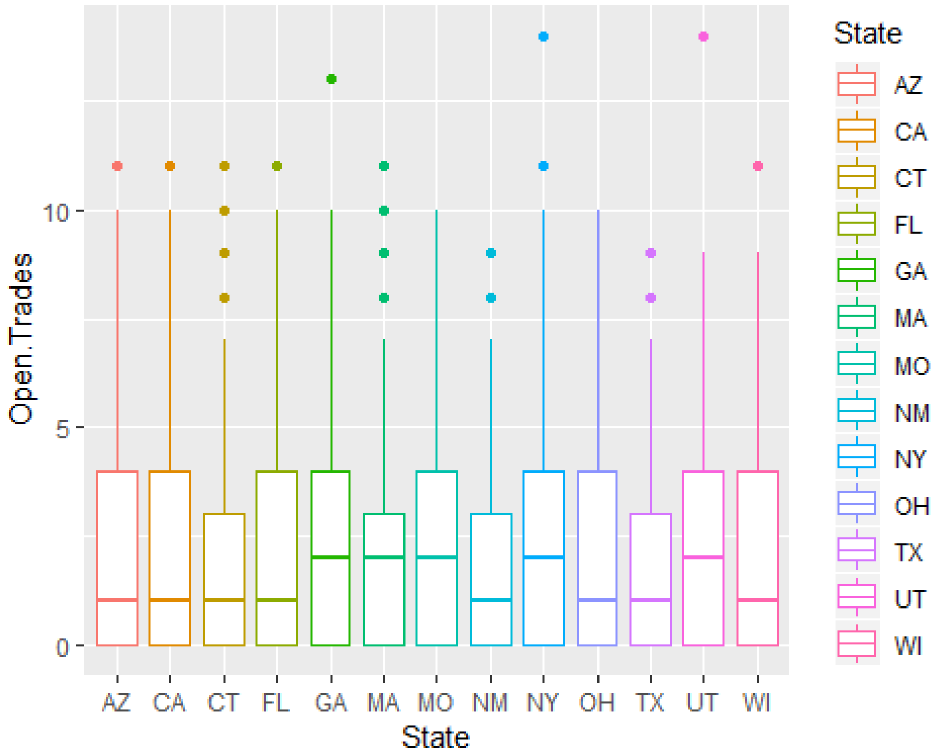



```
#Statistical summary of Open Trades:
describe(loan_data$Open.Trades)

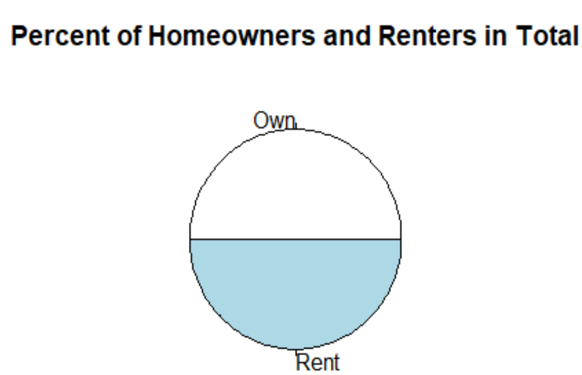
##    vars     n mean   sd median trimmed  mad min max range skew kurtosis
## X1    1 10000 2.07 2.22      1    1.75 1.48   0  14    14 0.99     0.43
##        se
## X1 0.02
```

```
# Box Plot of Open_Trade by States:
#The number of open trades vary widely across States with a big difference in
the average open trades and high standard deviation, which indicates a skewed
distribution.
#There are several extreme values in GA, NY and UT. Meanwhile CT and MA are t
wo states with the highest number of outliers of Open Trades.
loan_data %>%
```

```
        group_by(State) %>%
        ggplot(aes(x= State, y= Open.Trades, col = State)) +
        geom_boxplot()
```




```
#Question: What is overall proportion of "Own" and "Rent"?
# The pie chart illustrates an equal share of homeowners and renters in the l
oan_data set.
pie(table(loan_data$Home.Ownership),
    main = "Percent of Homeowners and Renters in Total")
```

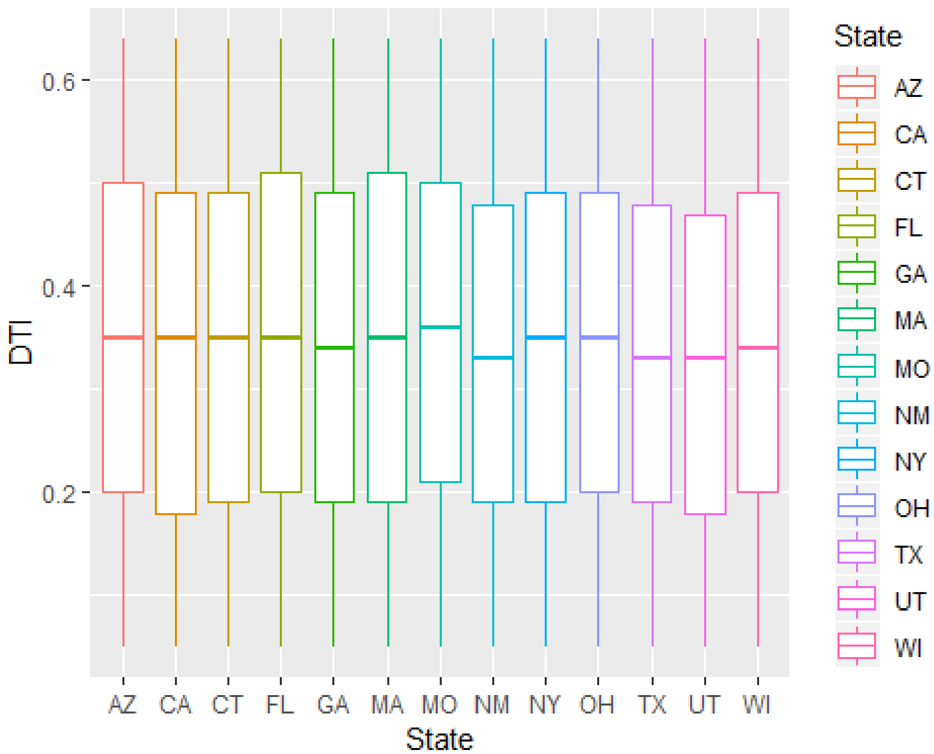


```
## Create Contingency Table for Two-level Homeownership for each State
# At state-level, the frequency homeownerhip is also equally divided between
"own" and 'rent'.
library(descr)
home_tab <- CrossTable(loan_data$State, loan_data$Home.Ownership, prop.c = FA
LSE, prop.chisq = FALSE, prop.t = FALSE)
home_tab
```

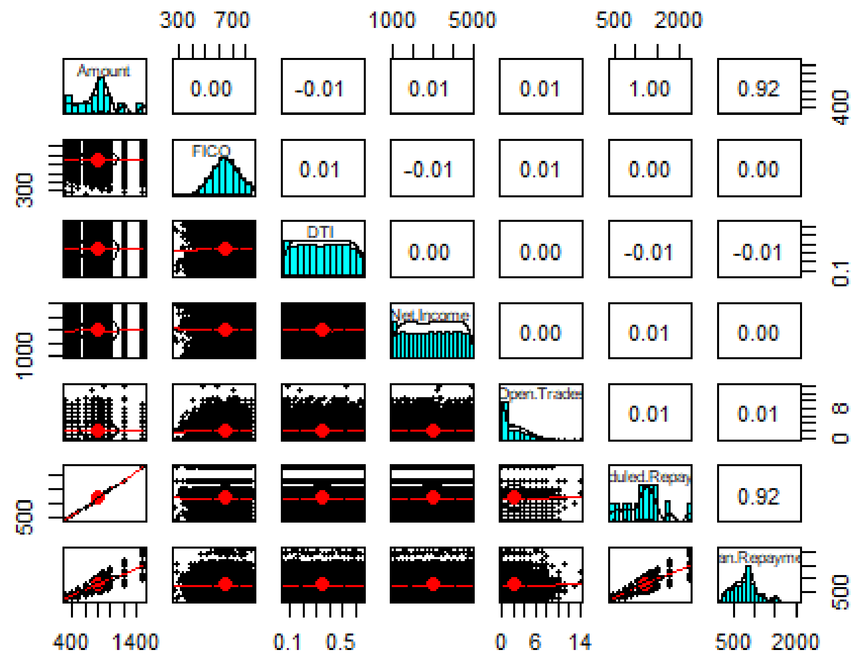
```
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |-------------------------|
##
## ========================================
##                      loan_data$Home.Ownership
## loan_data$State       Own     Rent    Total
## ----------------------------------------
## AZ                    384      357      741
##                     0.518    0.482    0.074
## ----------------------------------------
## CA                    412      382      794
##                     0.519    0.481    0.079
## ----------------------------------------
## CT                    394      393      787
##                     0.501    0.499    0.079
## ----------------------------------------
## FL                    394      433      827
##                     0.476    0.524    0.083
## ----------------------------------------
## GA                    385      392      777
##                     0.495    0.505    0.078
## ----------------------------------------
## MA                    372      393      765
##                     0.486    0.514    0.076
## ----------------------------------------
## MO                    409      386      795
##                     0.514    0.486    0.080
## ----------------------------------------
## NM                    355      386      741
##                     0.479    0.521    0.074
## ----------------------------------------
## NY                    359      404      763
##                     0.471    0.529    0.076
## ----------------------------------------
## OH                    380      354      734
##                     0.518    0.482    0.073
## ----------------------------------------
## TX                    404      390      794
##                     0.509    0.491    0.079
```

```
## ----------------------------------------
## UT                    364     357     721
##                     0.505   0.495   0.072
## ----------------------------------------
## WI                    393     368     761
##                     0.516   0.484   0.076
## ----------------------------------------
## Total                5005    4995   10000
## ========================================

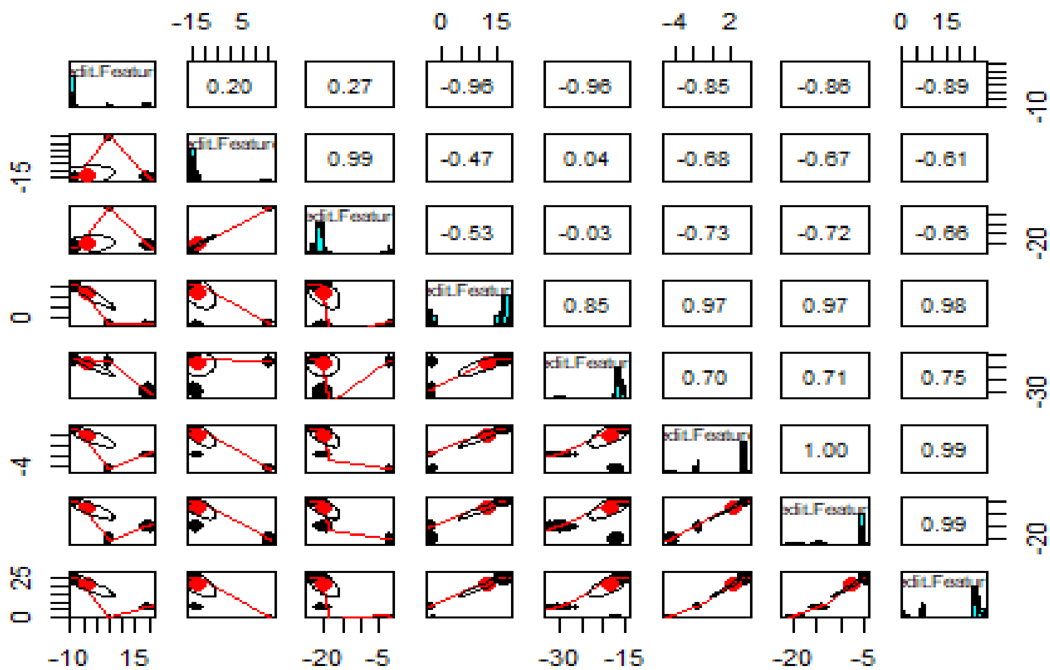
# Box Plot of DTI by States
# DTI has similiar distribution across States with very slight difference in
mean value
loan_data %>%
         group_by(State) %>%
         ggplot(aes(x= State, y= DTI, col = State)) +
         geom_boxplot()
```




```
# Question: What are the correlation among numeric variables? Is there multic
ollinearity issue?
#Amount, Loan.Repayment and Scheduled Repayment are nearly perfect correlated
with the coefficients at 0.92
pairs.panels(loan_data[c(2,3,4,5,7,8,9)])
```




```
# Question: What are the correlation among Credit Features 1-8? Is there multicollinearity issues?
# Feature 2 and 3, Feature 6 and 7, Feature 7 and 8 are perfectly correlated at 0.99 and 1. When building model, it is necessary to drop one out of the two variables to eliminate the multi-collinearity problem and make model stable.
pairs.panels(loan_data[c(10:17)])
```



III. Model Buiding: ## Regression Methods (Linear, Lasso, Elastic Net) for Predicting Risk ratio (Continous Target Variable)

```
## Data Preparation for modeling building:
# Remove variables that are not needed for the model: State,
data_1 <- loan_data[-1]

#Recode Categorical Variable, HomeOwnership into numeric value: 1 - Own, 0- R
ent
data_1$Home.Recode <- data_1$Home.Ownership %>% recode("Own" = 1, "Rent" = 0)

# Remove "Home.Ownership" column:
data_1$Home.Ownership <- NULL

# Derive a risk variable, continous response variable:
data_2 <- data_1 %>%
  mutate(Numeric.Response = Loan.Repayment/Scheduled.Repayment)

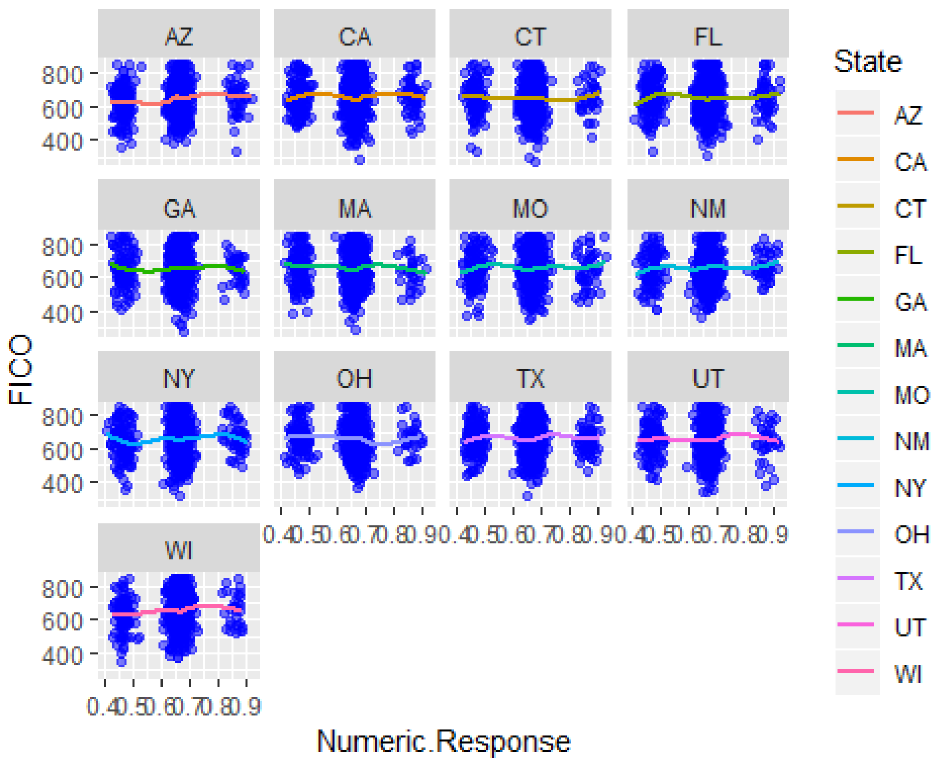
# Remove "Loan.Repayment" column from modeling dataset:
data_2$Loan.Repayment <- NULL

# Question: Is there any correlation between derived Numeric Targeted variabl
e and FICO score at State-level?

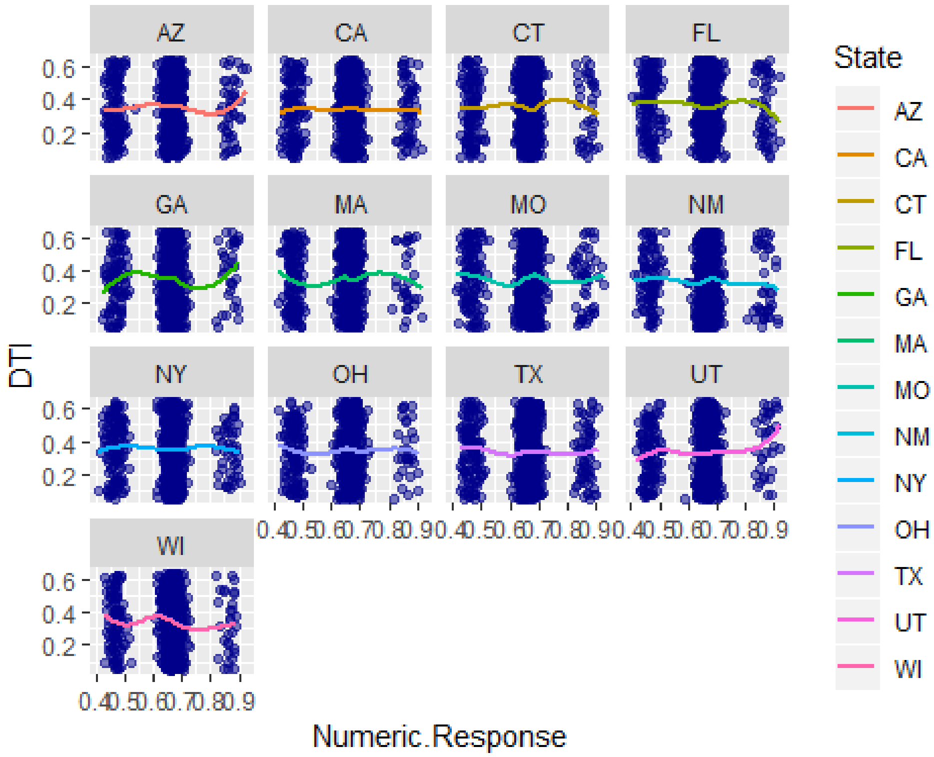

 #There is no association between Numeric Response variable and FICO at all S
```

```
tates.
data_3 <- loan_data %>%
  mutate(Numeric.Response = Loan.Repayment/Scheduled.Repayment)

  data_3 %>%
  group_by(State) %>%
         ggplot(aes(x= Numeric.Response, y = FICO, col=State)) +
         geom_point(alpha=0.5, color='blue') +
         geom_smooth(se=0, method = "loess") +
        facet_wrap(~State)
```




```
# Question: Is the DTI and Numeric Response correlated at the State-level?
 #There is no association between Numeric Response variable and DTI at States
-level.
data_3 %>%
group_by(State) %>%
         ggplot(aes(x= Numeric.Response, y = DTI, col=State)) +
         geom_point(alpha=0.5, color='darkblue') +
         geom_smooth(se=0, method = "loess") +
        facet_wrap(~State)
```




```
# Data Partition
set.seed(222)
ind <- sample(2, nrow(data_2), replace = T, prob = c(0.7, 0.3))
train <- data_2[ind==1,]
test <- data_2[ind==2,]

# Custom Control Parameters
custom <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 5,
                       verboseIter = T)

# Linear Model
set.seed(1234)
lm <- train(Numeric.Response ~ .,
            train,
            method = 'lm',
            trControl = custom)

## + Fold01.Rep1: intercept=TRUE
## - Fold01.Rep1: intercept=TRUE
## + Fold02.Rep1: intercept=TRUE
## - Fold02.Rep1: intercept=TRUE
## + Fold03.Rep1: intercept=TRUE
## - Fold03.Rep1: intercept=TRUE
## + Fold04.Rep1: intercept=TRUE
## - Fold04.Rep1: intercept=TRUE
```

```
## + Fold05.Rep1: intercept=TRUE
## - Fold05.Rep1: intercept=TRUE
## + Fold06.Rep1: intercept=TRUE
## - Fold06.Rep1: intercept=TRUE
## + Fold07.Rep1: intercept=TRUE
## - Fold07.Rep1: intercept=TRUE
## + Fold08.Rep1: intercept=TRUE
## - Fold08.Rep1: intercept=TRUE
## + Fold09.Rep1: intercept=TRUE
## - Fold09.Rep1: intercept=TRUE
## + Fold10.Rep1: intercept=TRUE
## - Fold10.Rep1: intercept=TRUE
```

``````

```
## Aggregating results
## Fitting final model on full training set
```

```
lm$results
```

```
##   intercept       RMSE  Rsquared         MAE       RMSESD  RsquaredSD
## 1      TRUE 0.02000799 0.9526773 0.01603586 0.0004153789 0.003820648
##         MAESD
## 1 0.000318481
```

```
# Summary of Linear Regression Results - Model outputs
lm
```

```
## Linear Regression
##
## 7036 samples
##   32 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 6332, 6333, 6333, 6333, 6333, 6332, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.02000799  0.9526773  0.01603586
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

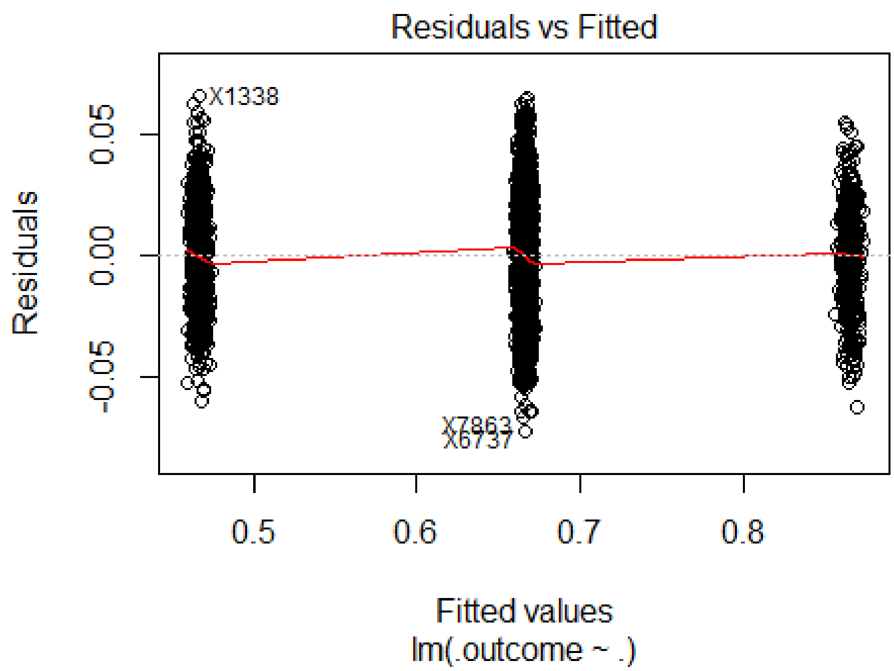
```
summary(lm)
```

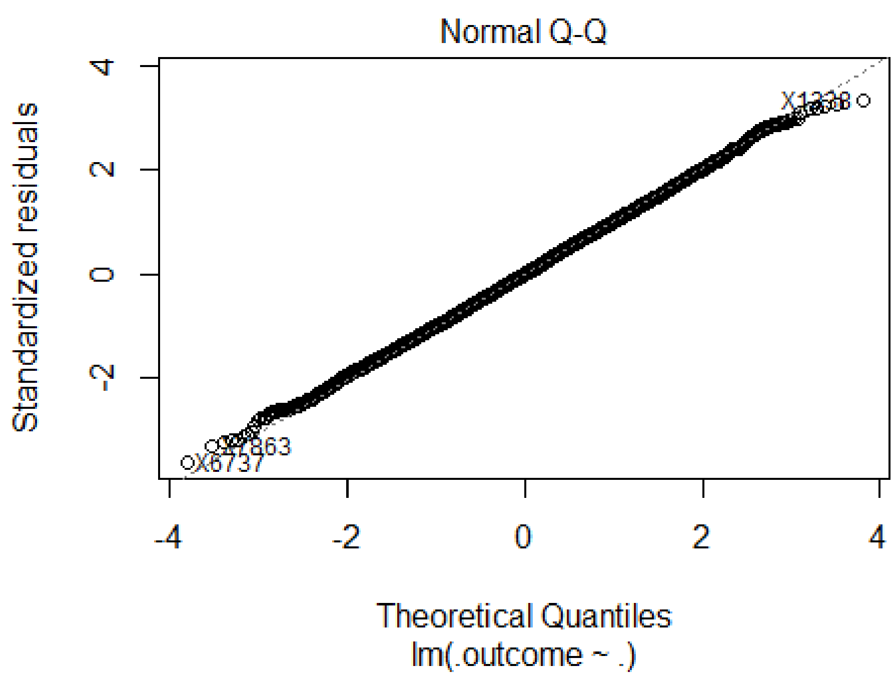
```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.072159 -0.013783 -0.000089  0.013695  0.066112
```
```

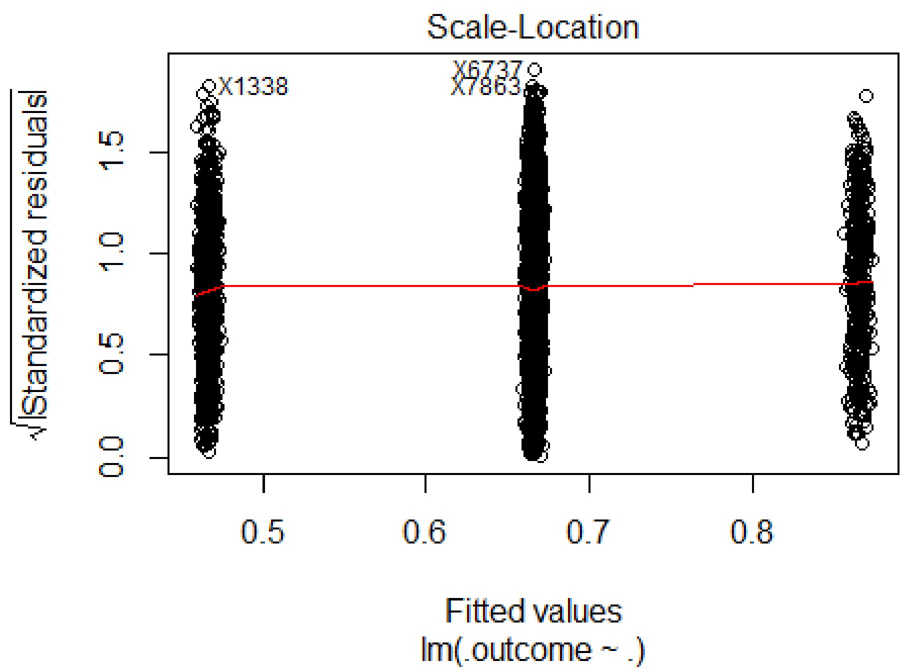
```
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           6.447e-01  2.453e-02  26.288  < 2e-16 ***
## Amount               -5.310e-04  2.141e-03  -0.248 0.804149
## FICO                  2.969e-06  2.407e-06   1.233 0.217522
## DTI                   2.924e-04  1.386e-03   0.211 0.832861
## Net.Income           -3.183e-07  2.075e-07  -1.534 0.125136
## Open.Trades          -1.129e-04  1.069e-04  -1.056 0.290927
## Scheduled.Repayment   3.546e-04  1.427e-03   0.248 0.803836
## Credit.Feature.1     -8.001e-04  6.914e-04  -1.157 0.247193
## Credit.Feature.2      8.217e-04  5.153e-04   1.595 0.110867
## Credit.Feature.3      9.416e-05  3.713e-04   0.254 0.799783
## Credit.Feature.4      7.136e-04  5.277e-04   1.352 0.176353
## Credit.Feature.5     -2.816e-04  4.131e-04  -0.682 0.495437
## Credit.Feature.6     -2.156e-03  2.384e-03  -0.904 0.365862
## Credit.Feature.7     -3.952e-04  8.790e-04  -0.450 0.653009
## Credit.Feature.8     -3.517e-04  3.464e-04  -1.015 0.309994
## Credit.Feature.9      9.269e-04  6.947e-04   1.334 0.182169
## Credit.Feature.10    -2.569e-03  7.870e-04  -3.264 0.001103 **
## Credit.Feature.11    -1.199e-03  5.611e-04  -2.136 0.032696 *
## Credit.Feature.12    -1.381e-03  6.794e-04  -2.032 0.042180 *
## Credit.Feature.13    -9.324e-05  1.120e-03  -0.083 0.933631
## Credit.Feature.14     2.205e-04  4.004e-04   0.551 0.581960
## Credit.Feature.15    -1.212e-04  1.746e-04  -0.694 0.487788
## Credit.Feature.16    -1.253e-03  4.661e-04  -2.688 0.007201 **
## Credit.Feature.17    -2.085e-04  3.020e-04  -0.690 0.490031
## Credit.Feature.18    -1.798e-03  4.502e-04  -3.995 6.54e-05 ***
## Credit.Feature.19    -2.517e-04  3.023e-04  -0.833 0.405119
## Credit.Feature.20    -2.242e-03  1.727e-03  -1.298 0.194186
## Credit.Feature.21     7.800e-04  1.090e-03   0.716 0.474315
## Credit.Feature.22    -1.911e-04  7.023e-04  -0.272 0.785541
## Credit.Feature.23    -8.892e-04  3.945e-04  -2.254 0.024221 *
## Credit.Feature.24    -1.290e-03  3.650e-04  -3.535 0.000411 ***
## Credit.Feature.25     4.256e-04  9.197e-04   0.463 0.643536
## Home.Recode           3.580e-04  4.769e-04   0.751 0.452835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01997 on 7003 degrees of freedom
## Multiple R-squared:  0.9532, Adjusted R-squared:  0.953
## F-statistic:  4458 on 32 and 7003 DF,  p-value: < 2.2e-16

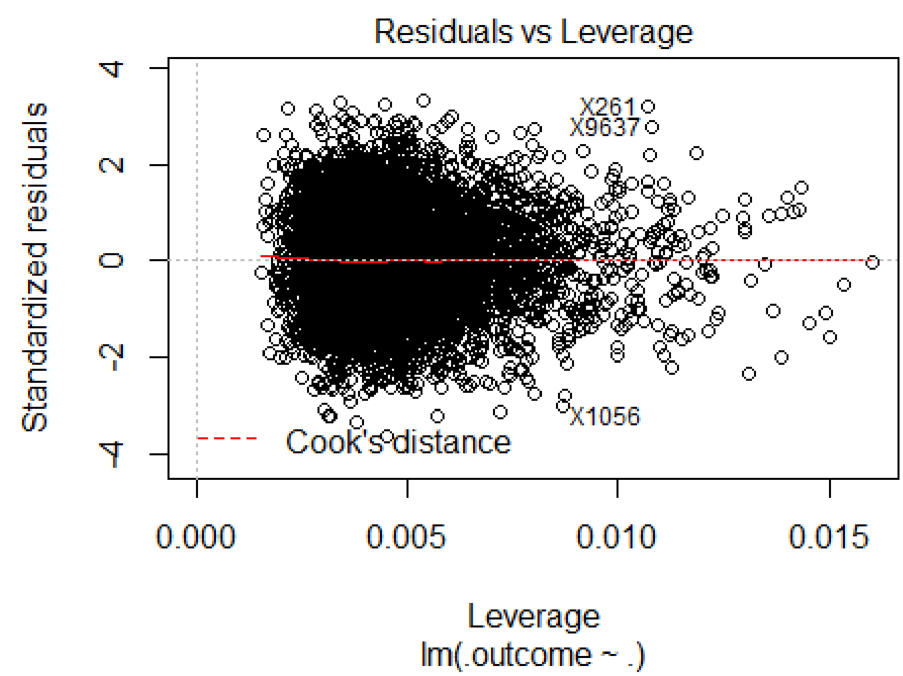
#Plot Residuals vs Fitted vs. Leverage, Standardized Residuals,

plot(lm$finalModel)
```


Residuals vs Fitted
Residuals
0.05
0.00
-0.05
X1338
X7863
X6737
0.5
0.6
0.7
0.8
Fitted values
lm(.outcome ~ .)


Normal Q-Q
Standardized residuals
4
2
0
-2
X1338
X7863
X6737
-4
-2
0
2
4
Theoretical Quantiles
lm(.outcome ~ .)




```
# Lasso Regression (L1 Penalty)

set.seed(1234)
lasso <- train(Numeric.Response ~.,
```

```r
               train,
               method = 'glmnet',
               tuneGrid = expand.grid(alpha = 1,
                                      lambda = seq(0.001, 1, length = 5)),
               trControl = custom)
```

```
## + Fold01.Rep1: alpha=1, lambda=1
## - Fold01.Rep1: alpha=1, lambda=1
## + Fold02.Rep1: alpha=1, lambda=1
## - Fold02.Rep1: alpha=1, lambda=1
## + Fold03.Rep1: alpha=1, lambda=1
## - Fold03.Rep1: alpha=1, lambda=1
## + Fold04.Rep1: alpha=1, lambda=1
## - Fold04.Rep1: alpha=1, lambda=1
## + Fold05.Rep1: alpha=1, lambda=1
## - Fold05.Rep1: alpha=1, lambda=1
## + Fold06.Rep1: alpha=1, lambda=1
## - Fold06.Rep1: alpha=1, lambda=1
## + Fold07.Rep1: alpha=1, lambda=1
## - Fold07.Rep1: alpha=1, lambda=1
## + Fold08.Rep1: alpha=1, lambda=1
## - Fold08.Rep1: alpha=1, lambda=1
## + Fold09.Rep1: alpha=1, lambda=1
## - Fold09.Rep1: alpha=1, lambda=1
## + Fold10.Rep1: alpha=1, lambda=1
## - Fold10.Rep1: alpha=1, lambda=1
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

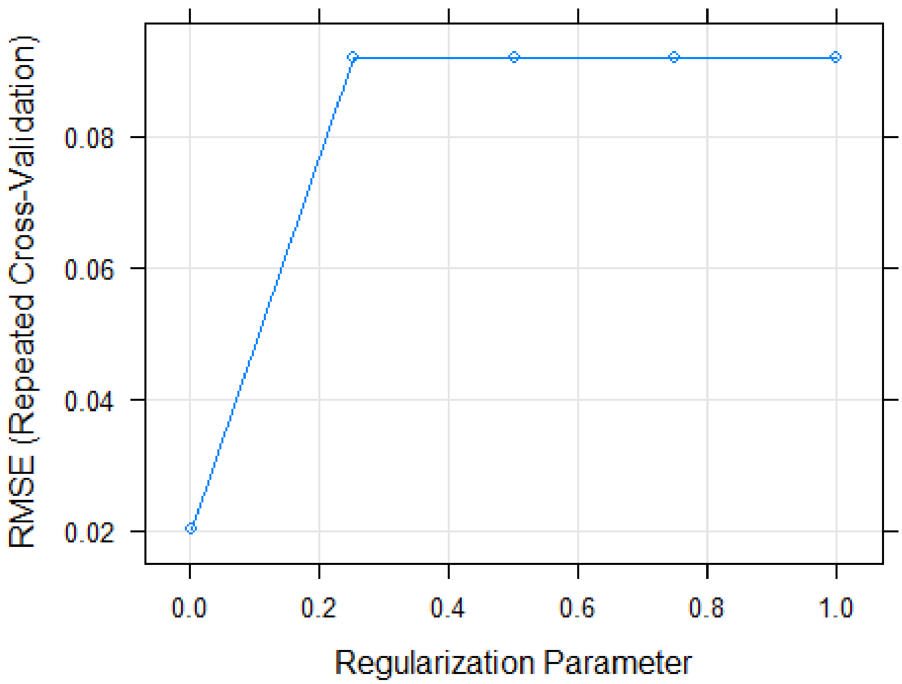
```
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 1, lambda = 0.001 on full training set
```

```r
# Summary of Lasso Regression's Result:
lasso
```

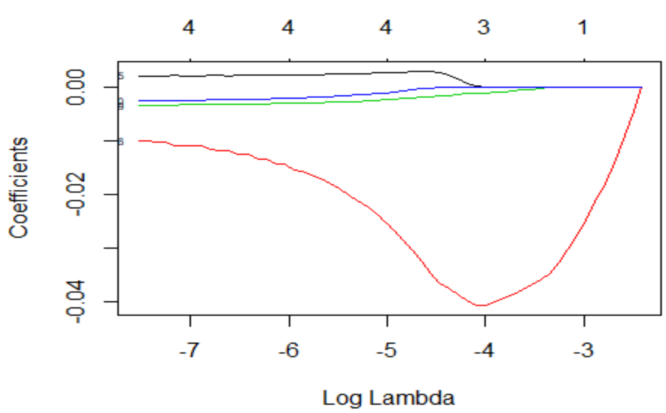
```
## glmnet
##
## 7036 samples
##   32 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 6332, 6333, 6333, 6333, 6333, 6332, ...
## Resampling results across tuning parameters:
##
##   lambda   RMSE        Rsquared   MAE
##   0.00100  0.02012988  0.9522277  0.01614918
##   0.25075  0.09205294        NaN  0.05835627
##   0.50050  0.09205294        NaN  0.05835627
##   0.75025  0.09205294        NaN  0.05835627
```

```
##    1.00000  0.09205294        NaN  0.05835627
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 0.001.
```

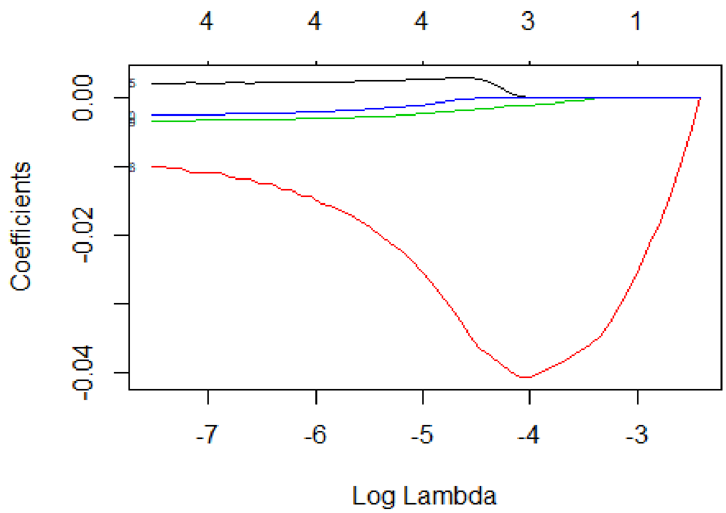
```r
# Plot Lasso Regression's Result
plot(lasso)
```



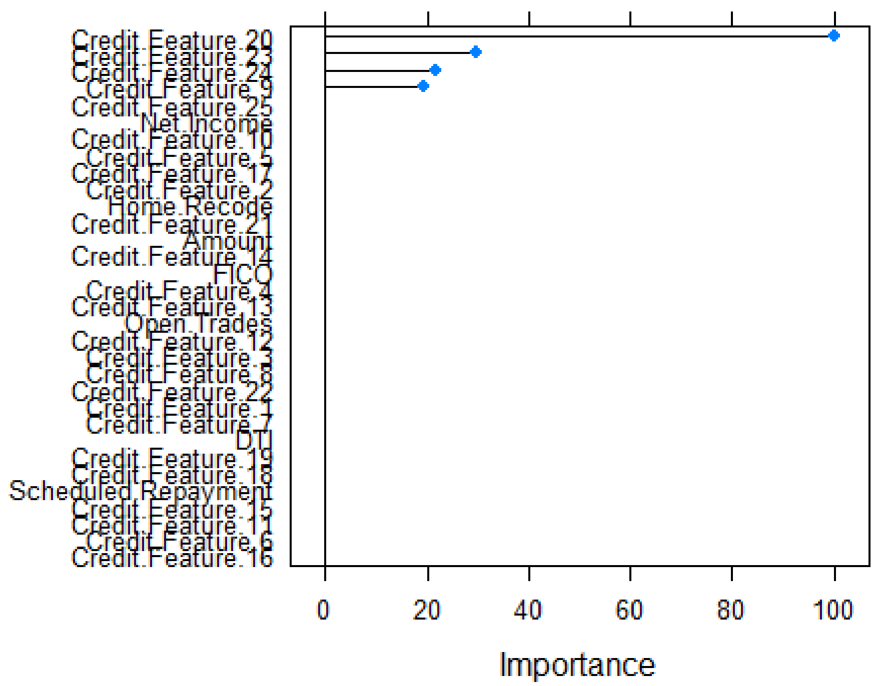

```r
plot(lasso$finalModel, xvar = 'lambda', label=T)
```




```r
# Plot Lasso Regression's Result
plot(lasso$finalModel, xvar = 'lambda', label=T)
```

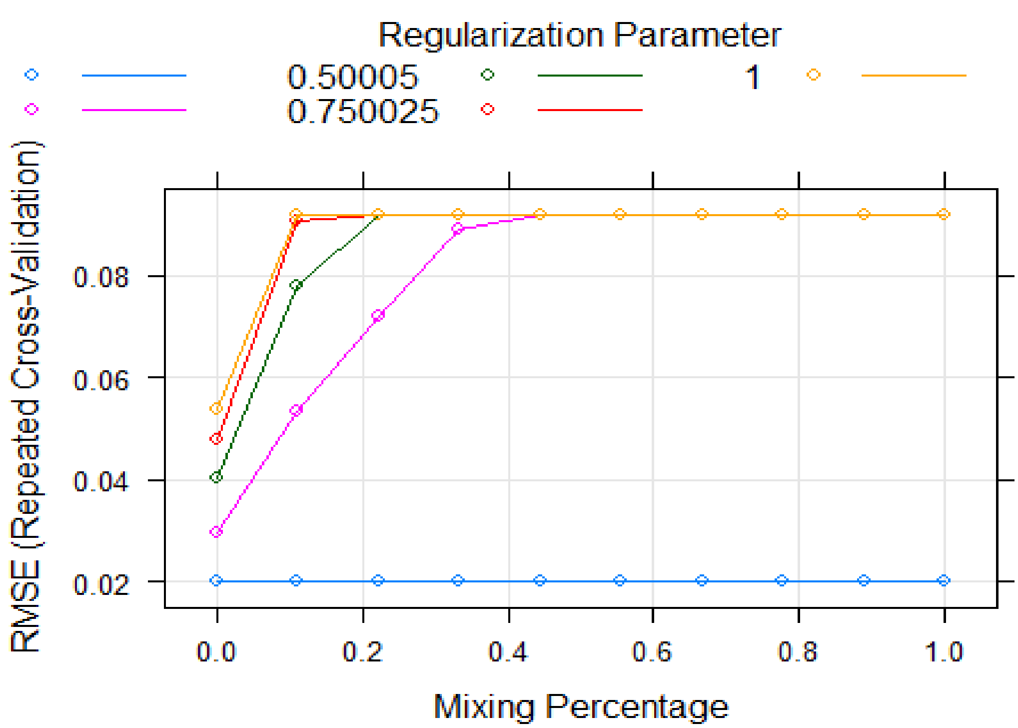



```
# Plot variable importance in lasso regression
plot(varImp(lasso, scale=T))
```

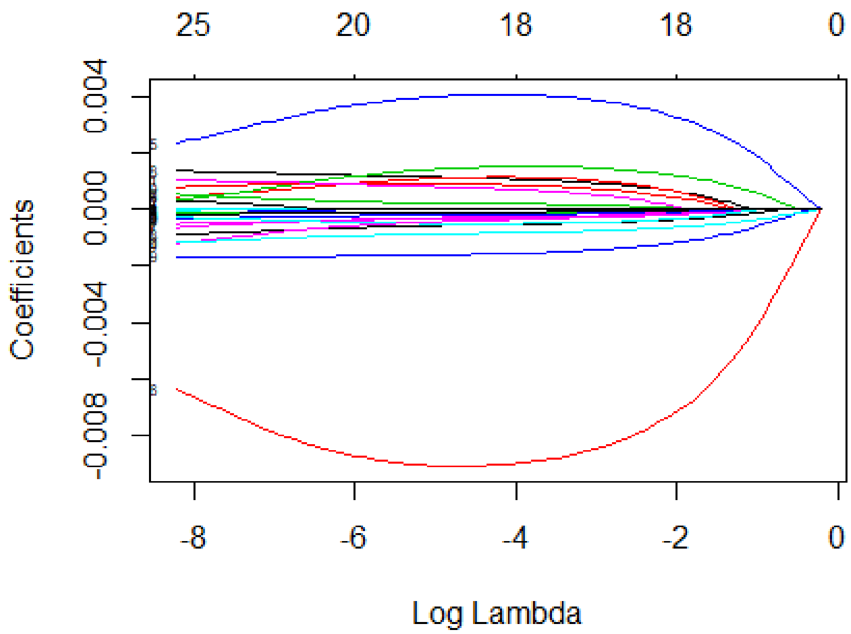



```
# Elastic Net Regression
set.seed(1234)
en <- train(Numeric.Response ~ .,
            train,
            method = 'glmnet',
            tuneGrid = expand.grid(alpha = seq(0,1, length =10),
```

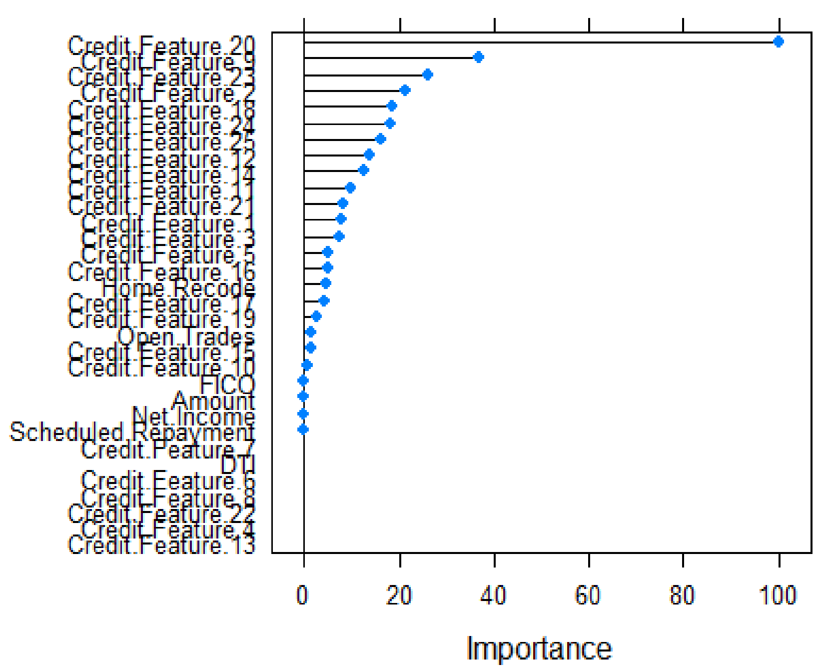
```
                                              lambda = seq(0.0001, 1, length = 5)),
            trControl = custom)
## + Fold01.Rep1: alpha=0.0000, lambda=1
## - Fold01.Rep1: alpha=0.0000, lambda=1
## + Fold01.Rep1: alpha=0.1111, lambda=1
## - Fold01.Rep1: alpha=0.1111, lambda=1
## + Fold01.Rep1: alpha=0.2222, lambda=1
## - Fold01.Rep1: alpha=0.2222, lambda=1
## + Fold01.Rep1: alpha=0.3333, lambda=1
## - Fold01.Rep1: alpha=0.3333, lambda=1
## + Fold01.Rep1: alpha=0.4444, lambda=1
## - Fold01.Rep1: alpha=0.4444, lambda=1
## + Fold01.Rep1: alpha=0.5556, lambda=1
## - Fold01.Rep1: alpha=0.5556, lambda=1
## + Fold01.Rep1: alpha=0.6667, lambda=1
## - Fold01.Rep1: alpha=0.6667, lambda=1
## + Fold01.Rep1: alpha=0.7778, lambda=1
## - Fold01.Rep1: alpha=0.7778, lambda=1
## + Fold01.Rep1: alpha=0.8889, lambda=1
## - Fold01.Rep1: alpha=0.8889, lambda=1
## + Fold01.Rep1: alpha=1.0000, lambda=1
## - Fold01.Rep1: alpha=1.0000, lambda=1
## + Fold02.Rep1: alpha=0.0000, lambda=1
## - Fold02.Rep1: alpha=0.0000, lambda=1
# Plot regularization parameter of Elastic Net:
plot(en)
```



```
# Plot Coefficients of Elastic Net Regression:
plot(en$finalModel, xvar = 'lambda', label=T)
```

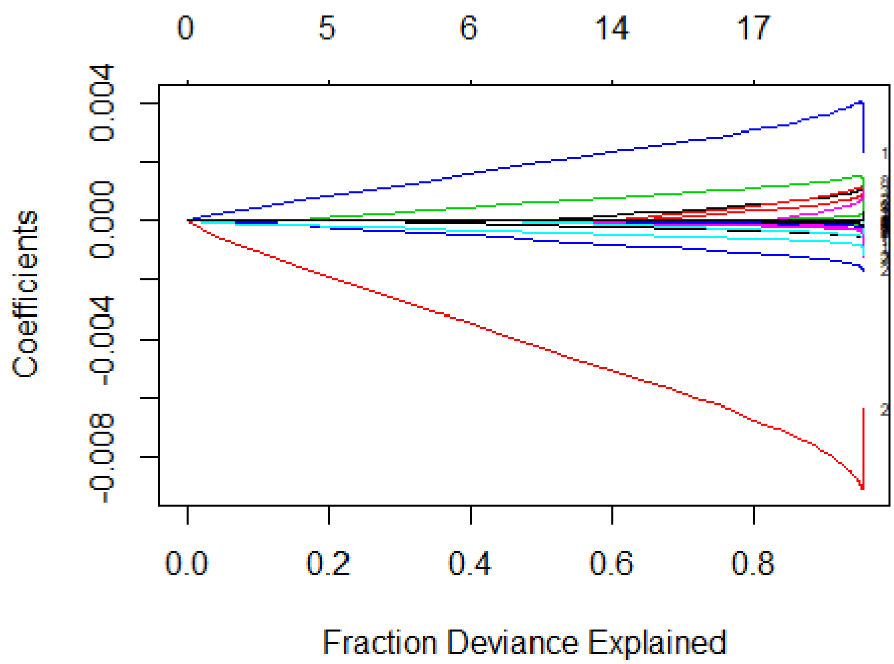



```
# Plot variable importance of Elastic Net Regression:
plot(varImp(en))
```



```r
# Credit Feature 20 serves as the most important predictor in the model

# Plot Fraction Deviance Explained of Elastic Net Regression:
plot(en$finalModel, xvar = 'dev', label=T)
```




```r
# Compare Models
model_list <- list(LinearModel = lm, Lasso = lasso, ElasticNet = en)
res <- resamples(model_list)
summary(res)

##
## Call:
## summary.resamples(object = res)
##
## Models: LinearModel, Lasso, ElasticNet
## Number of resamples: 50
##
## MAE
##                   Min.    1st Qu.     Median       Mean    3rd Qu.
## LinearModel 0.01519666 0.01579849 0.01603779 0.01603586 0.01620699
## Lasso       0.01534923 0.01591917 0.01607806 0.01614918 0.01639498
## ElasticNet  0.01524182 0.01580463 0.01599908 0.01604302 0.01624282
##                   Max. NA's
## LinearModel 0.01679698    0
## Lasso       0.01694383    0
## ElasticNet  0.01675403    0
##
```

```
## RMSE
##                   Min.    1st Qu.     Median       Mean    3rd Qu.
## LinearModel 0.01905455 0.01973747 0.01994562 0.02000799 0.02021463
## Lasso       0.01920980 0.01991923 0.02004691 0.02012988 0.02034345
## ElasticNet  0.01911573 0.01977980 0.01994264 0.02001636 0.02024623
##                   Max. NA's
## LinearModel 0.02101529    0
## Lasso       0.02130203    0
## ElasticNet  0.02102611    0
##
## Rsquared
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## LinearModel 0.9419913 0.9496225 0.9531156 0.9526773 0.9551590 0.9614449
## Lasso       0.9416286 0.9495865 0.9522910 0.9522277 0.9547737 0.9610770
## ElasticNet  0.9422877 0.9498059 0.9529515 0.9526399 0.9550465 0.9612055
##             NA's
## LinearModel    0
## Lasso          0
## ElasticNet     0
```

```
# Prediction error: Root Mean Square Error for Linear Model:
p1 <- predict(lm, train)
sqrt(mean((train$Numeric.Response-p1)^2))
```

```
## [1] 0.01991933
```

```
p2 <- predict(lm, test)
sqrt(mean((test$Numeric.Response-p2)^2))
```

```
## [1] 0.01998626
```

```
# Prediction error: Root Mean Square Err for Elastic Net Model:
p1 <- predict(en, train)
sqrt(mean((train$Numeric.Response-p1)^2))
```

```
## [1] 0.01997438
```

```
p2 <- predict(en, test)
sqrt(mean((test$Numeric.Response-p2)^2))
```

```
## [1] 0.02000775
```

```
# Prediction error: Root Mean Square Err for Lasso Model:
p1 <- predict(lasso, train)
sqrt(mean((train$Numeric.Response-p1)^2))
```

```
## [1] 0.02011953
```

```
p2 <- predict(lasso, test)
sqrt(mean((test$Numeric.Response-p2)^2))
```

```
## [1] 0.02012778
```

```
## Concluding Remarks on The Best Model:

# All models (Linear Regression, Lasso and Elastic Net) have similiar values
for key metrics (low RMSE, high adjusted Rsquare explaining variability of Nu
meric Response Variable). However, Linear Regression has the smaller predicti
on errors on both training and test set with a slight difference. In term of
computing resources, Linear Model is slightly better than Elastic Net and mor
e efficient. Therefore, the best prediction model for derived targed variable
in this case is Linear Regression Model.
```