

# Credit\_Default\_Prediction

Aishwarya

2023-04-06

## Loading required packages

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-6
```

```
library(caret)
```

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

```
## Loading required package: lattice
```

```
library(psych)
```

```
##
```

```
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
```

```
##
```

```
##      %+%, alpha
```

```
library(pls)
```

```
##
```

```
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
```

```
##
```

```
##      R2
```

```
## The following object is masked from 'package:stats':
```

```
##
```

```
##      loadings
```

## Data format pre-processing

```
train_default_data<-read.csv("loan_train_final.csv")
test_default_data<-read.csv("loan_test_final.csv")
```

```
#Employment column
train_default_data$employment<-as.integer(train_default_data$employment)
```

```
## Warning: NAs introduced by coercion
```

```
test_default_data$employment<-as.integer(test_default_data$employment)
```

```
## Warning: NAs introduced by coercion
```

```
#Train
sum(is.na(train_default_data$employment)) #333 NAs
```

```
## [1] 333
```

```
length(train_default_data$employment) #Out of 600
```

```
## [1] 2400
```

```
mean_value=mean(train_default_data$employment,na.rm=TRUE)
train_default_data$employment <- ifelse(is.na(train_default_data$employment), mean_value, train_default_data$employment)
```

```
#Test
sum(is.na(test_default_data$employment)) #63 NAs
```

```
## [1] 63
```

```
length(test_default_data$employment) #Out of 2400
```

```
## [1] 600
```

```
mean_value=mean(test_default_data$employment,na.rm=TRUE)
test_default_data$employment <- ifelse(is.na(test_default_data$employment), mean_value, test_default_data$employment)
```

### Term column processing

```
train_default_data$term <- as.numeric(gsub("yrs", "", train_default_data$term))
test_default_data$term <- as.numeric(gsub("yrs", "", test_default_data$term))
```

```
# Check for missing values
sum(is.na(train_default_data$term))
```

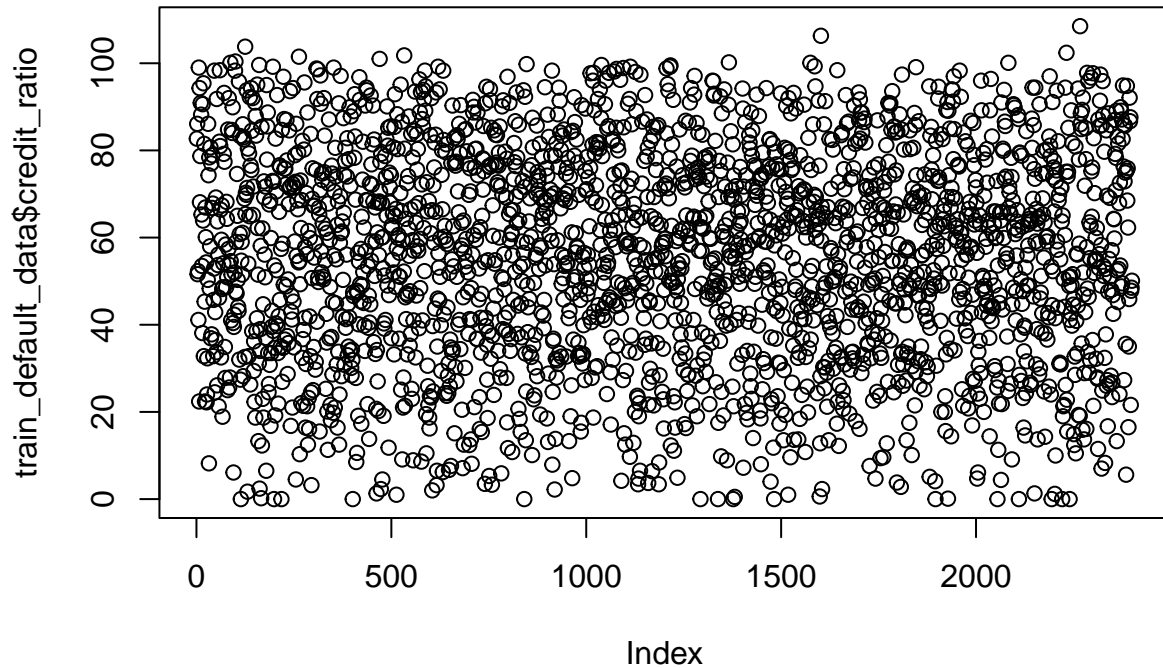
```
## [1] 0
```

```
sum(is.na(test_default_data))
```

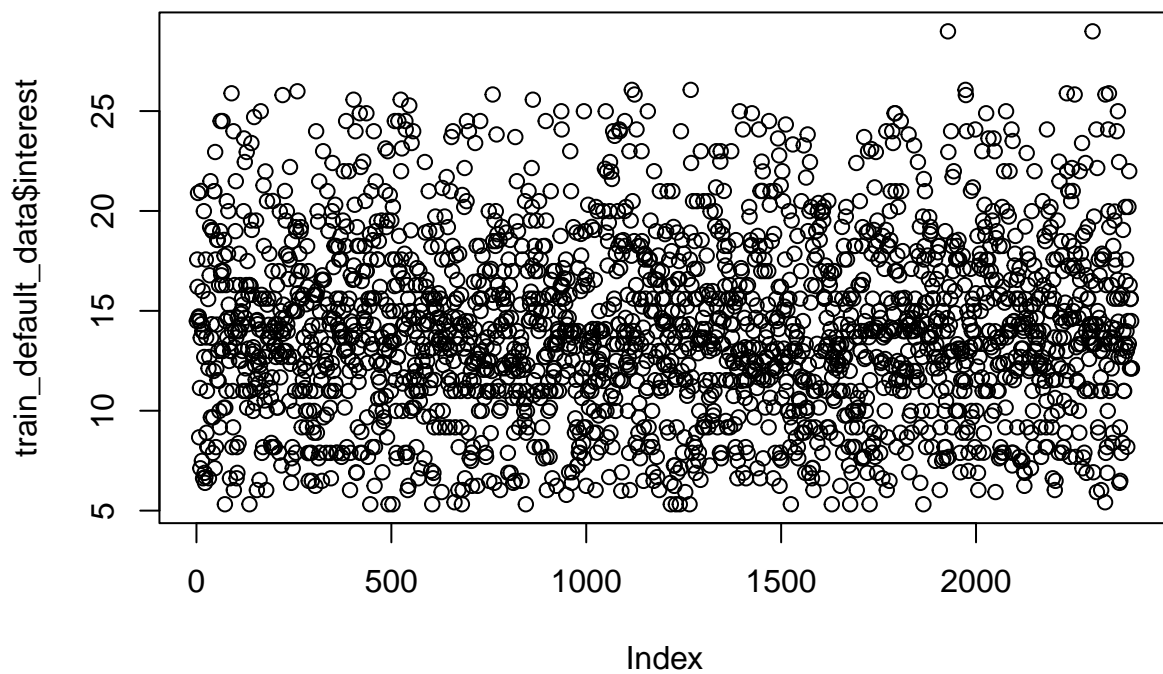
```
## [1] 0
```

```
# Create a random sample of 80% of the data for training  
library(rsample)  
train_sample <- initial_split(train_default_data, prop = 0.8)  
train_data <- training(train_sample)  
test_data <- testing(train_sample)
```

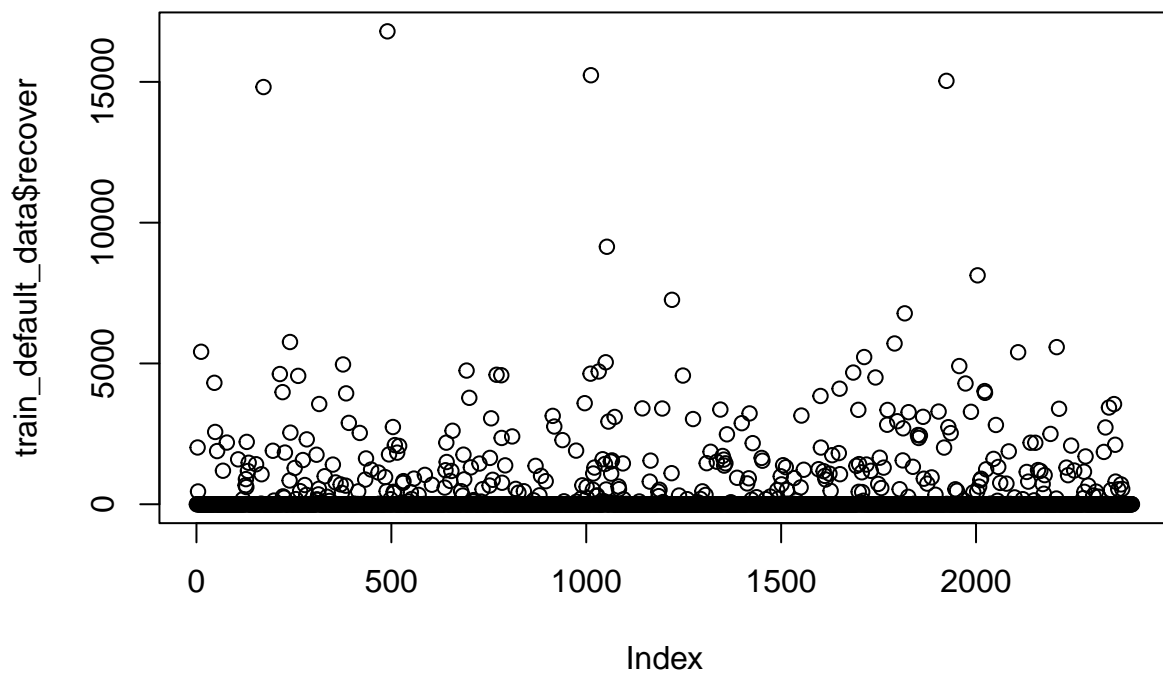
```
#EDA  
plot(train_default_data$credit_ratio)
```



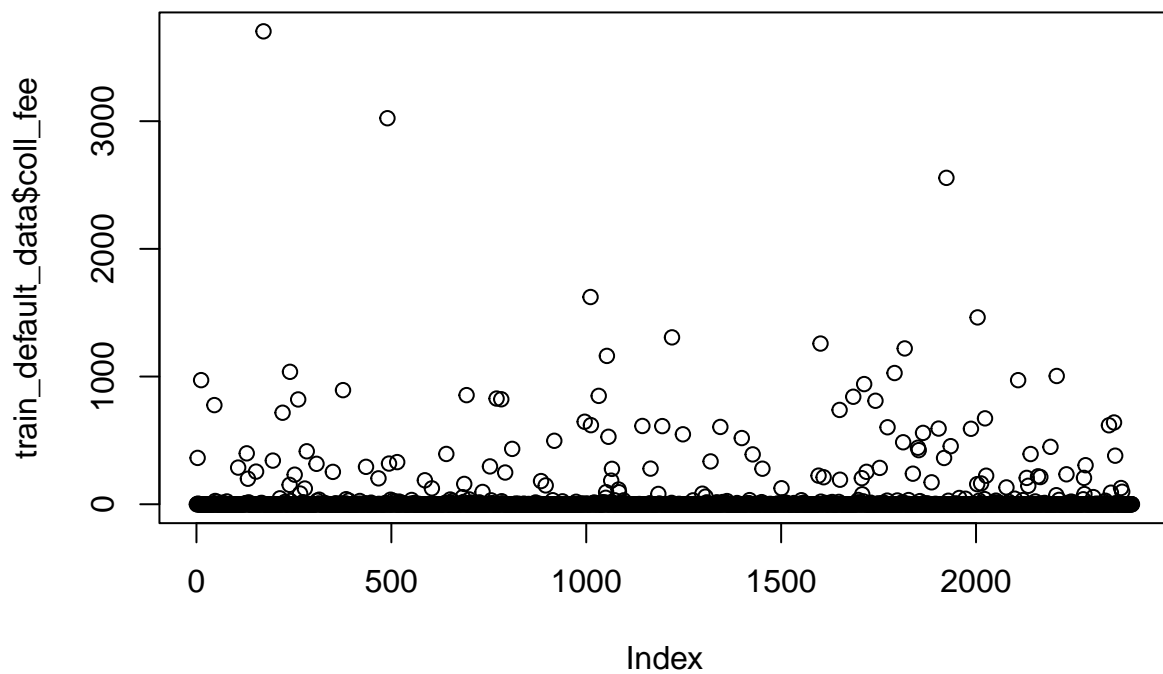
```
plot(train_default_data$interest)
```



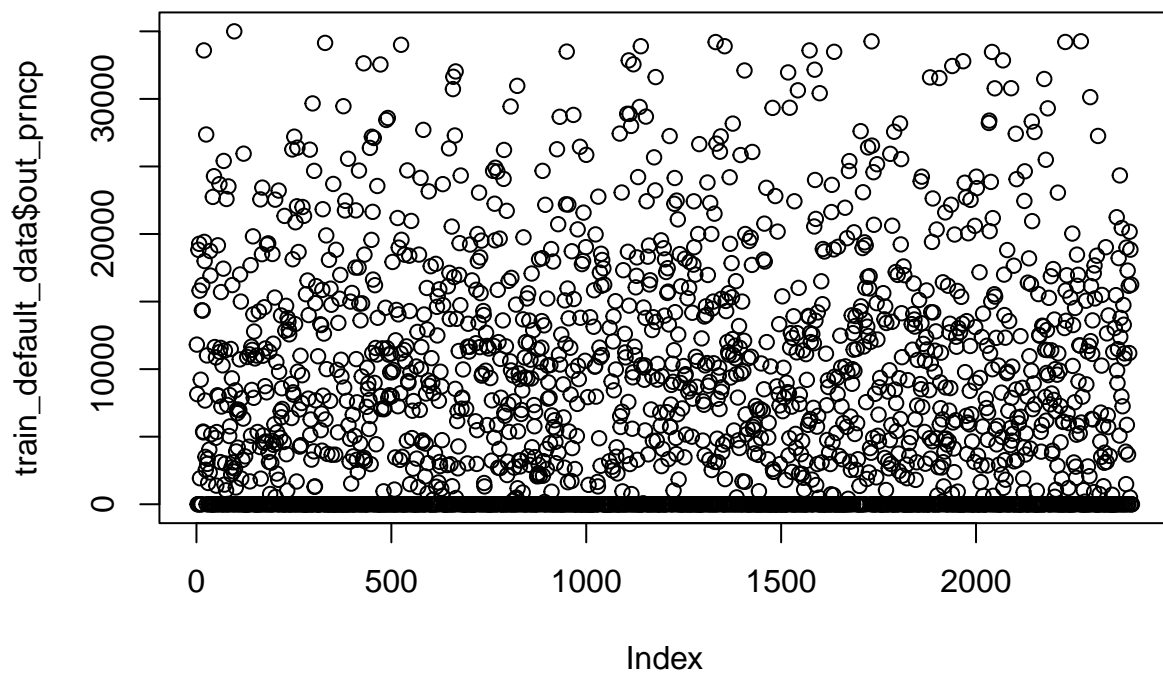
```
plot(train_default_data$recover)
```



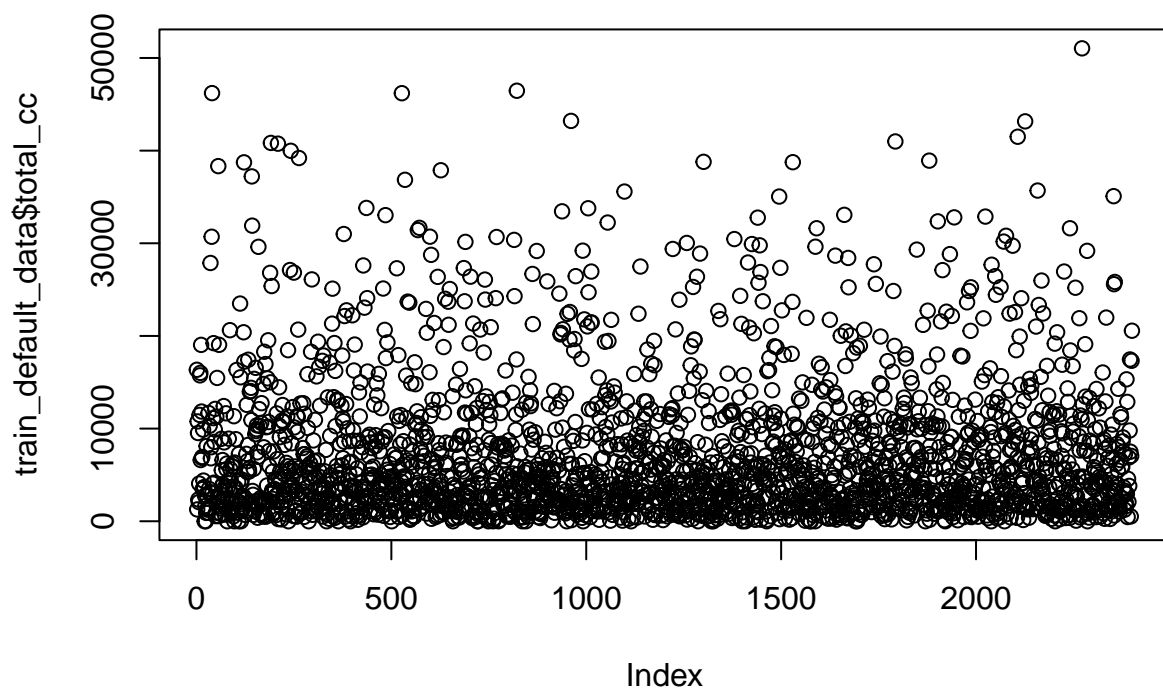
```
plot(train_default_data$coll_fee)
```



```
plot(train_default_data$out_prncp)
```

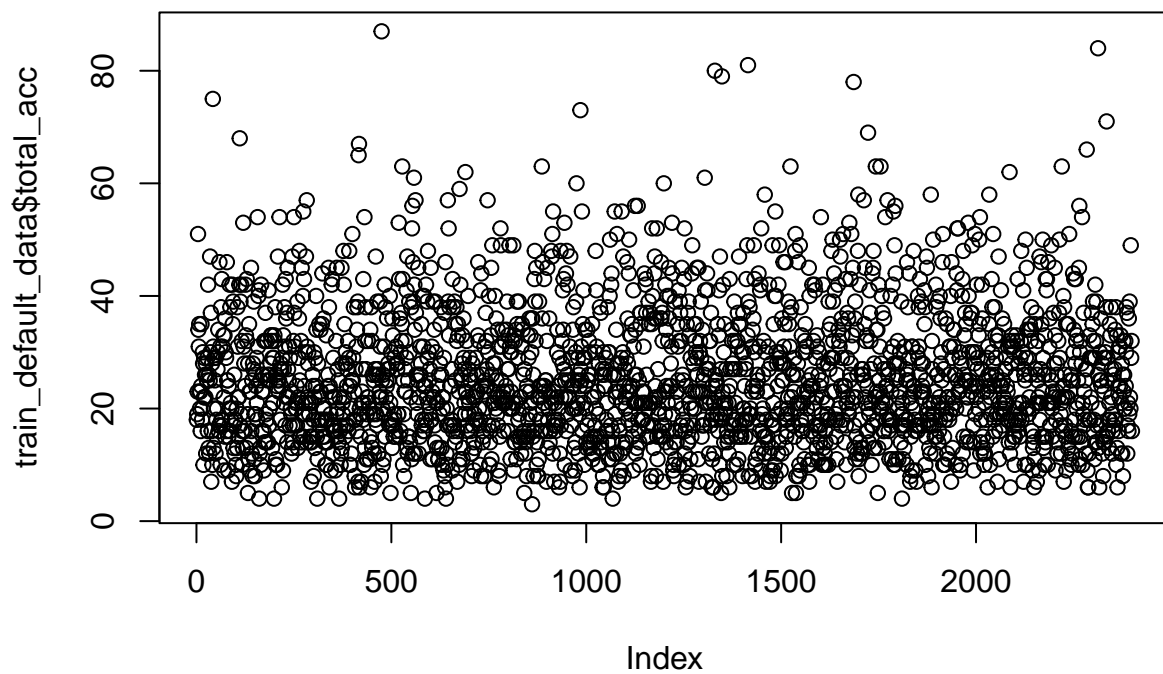


```
plot(train_default_data$total_cc)
```

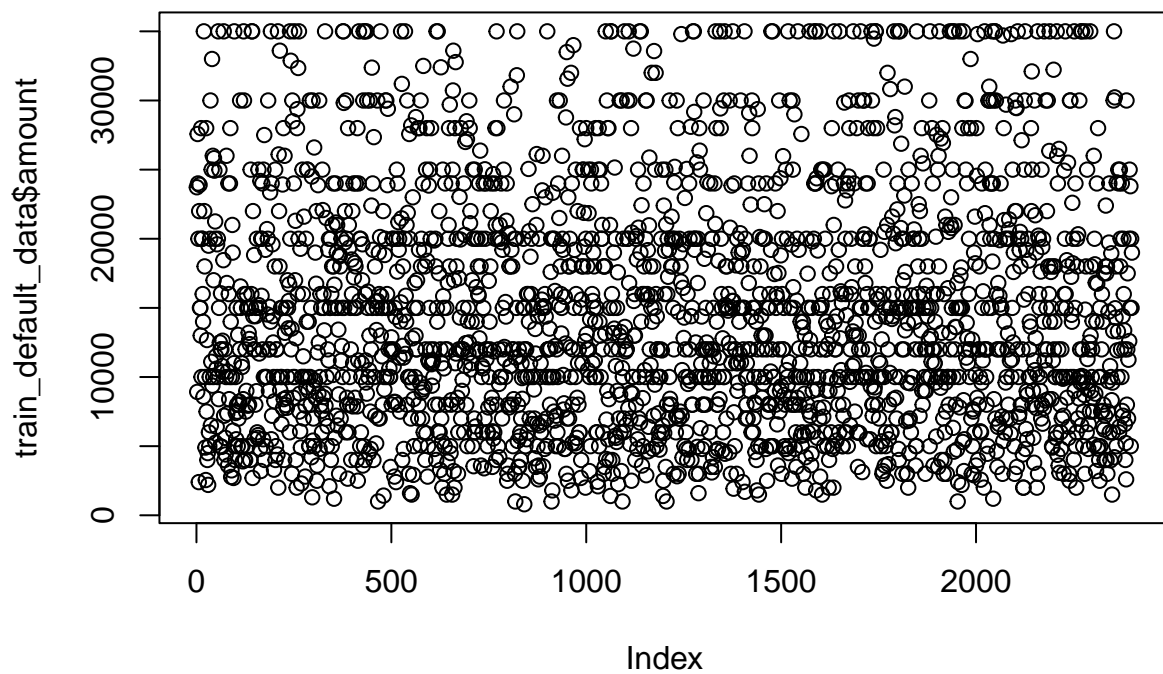


```
plot(train_default_data$total_acc)
```

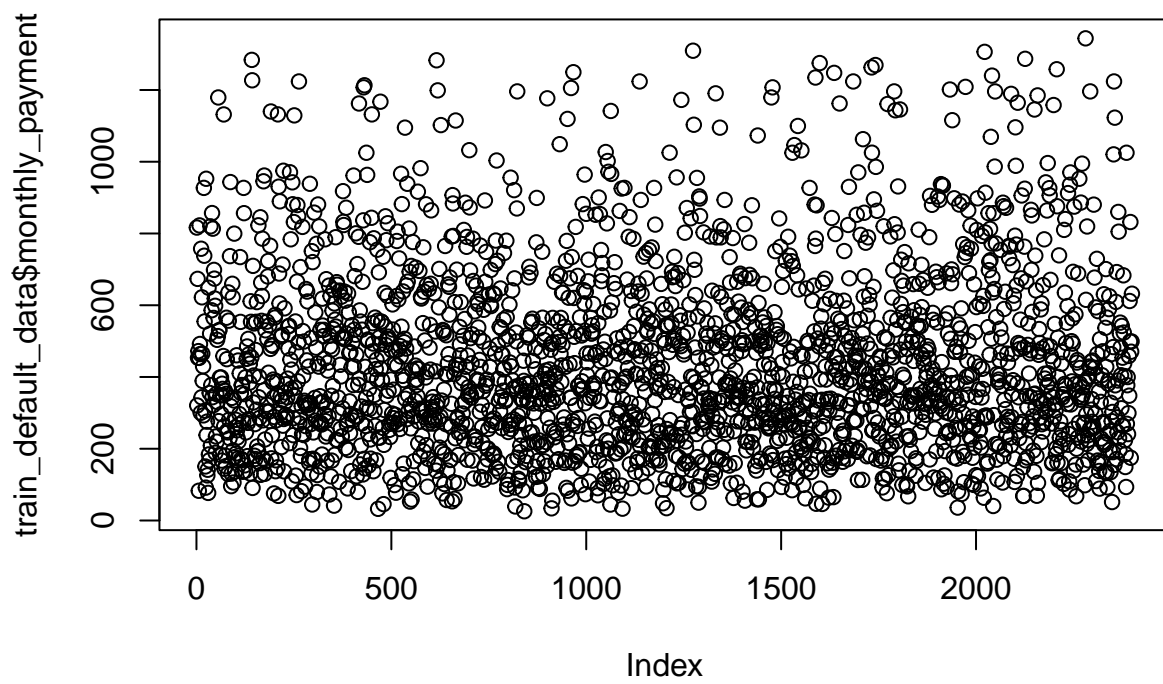




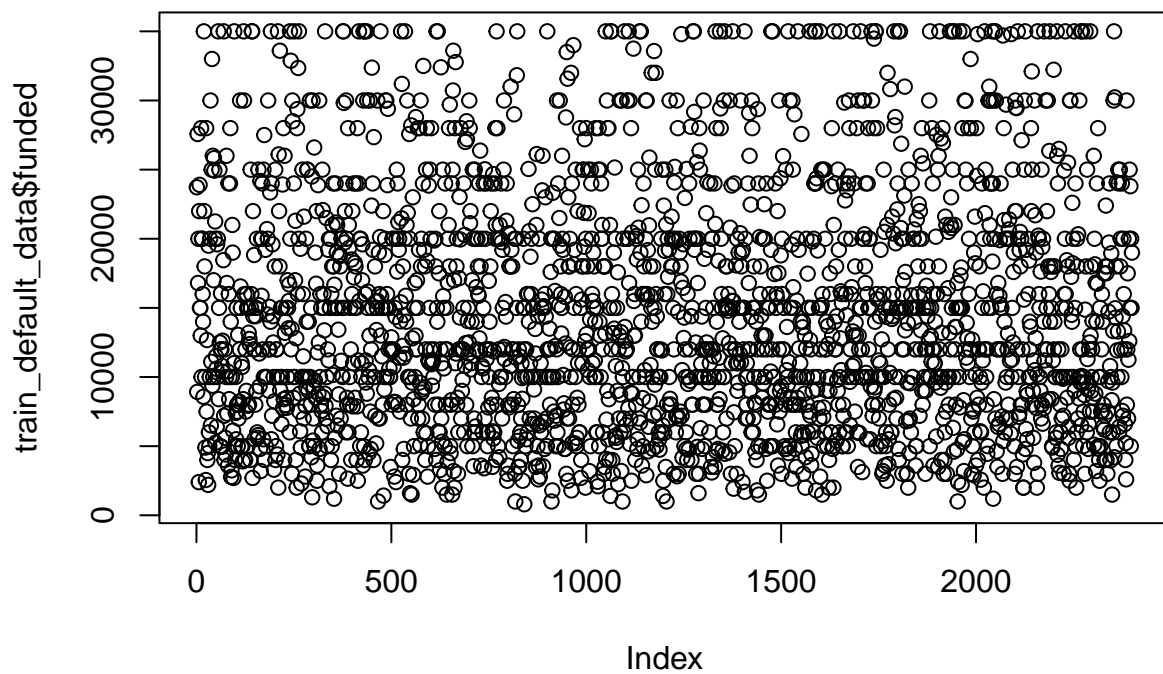
```
plot(train_default_data$amount)
```



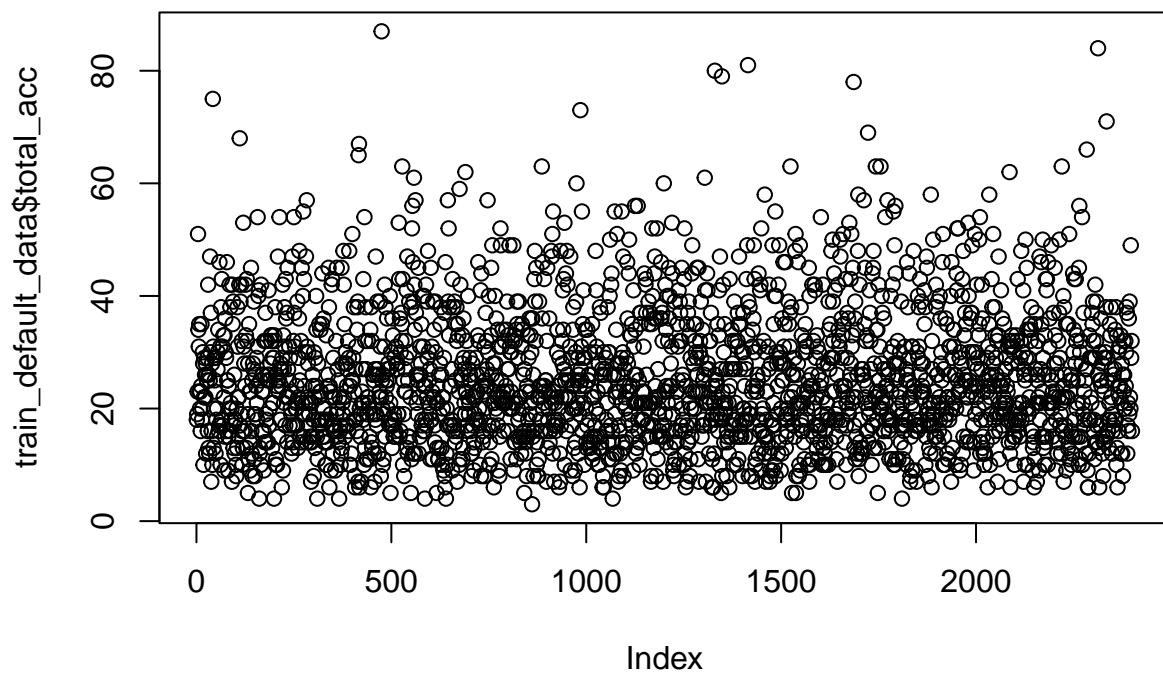
```
plot(train_default_data$monthly_payment)
```



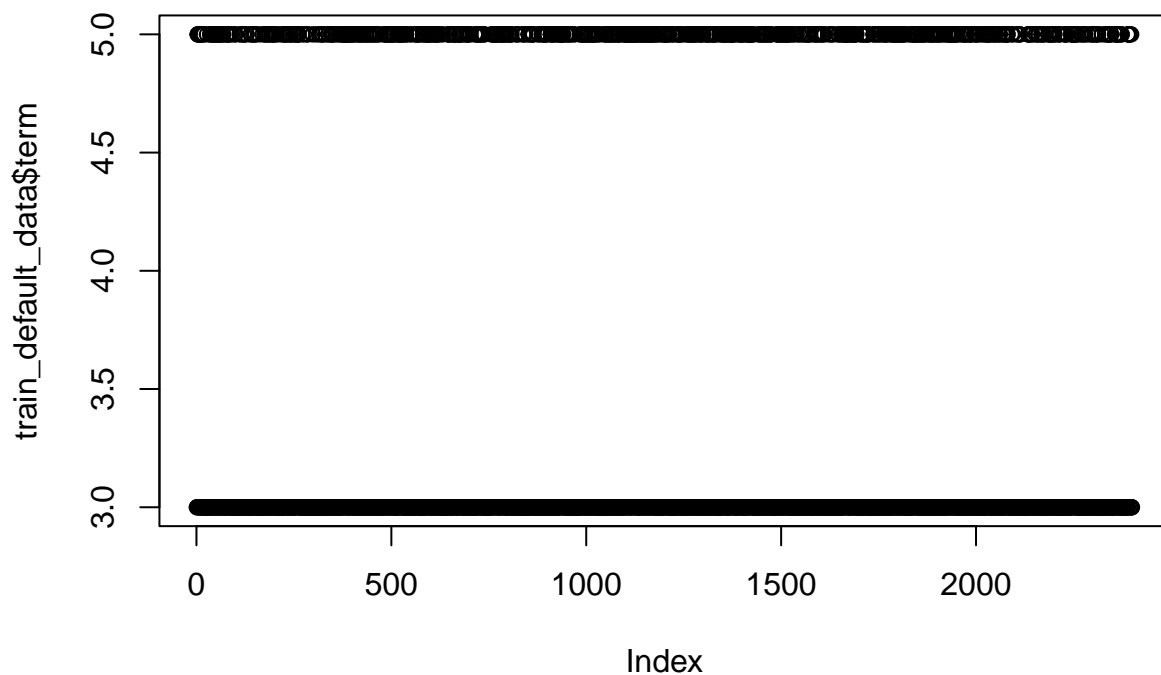
```
plot(train_default_data$funded)
```



```
plot(train_default_data$total_acc)
```



```
plot(train_default_data$term)
```



```
#Dataframe with all numerical variables
cor_data<-data.frame(train_default_data$credit_ratio,train_default_data$interest,train_default_data$recover,train_default_data$coll_fee,train_default_data$out_prncp,train_default_data$total_cc,train_default_data$total_acc,train_default_data$amount,train_default_data$monthly_payment,train_default_data$funded,train_default_data$total_acc.1,train_default_data$term)

# pairs(cor_data, pch = 19)

#Correlation
cor(cor_data)
```

```
##                                train_default_data.credit_ratio
## train_default_data.credit_ratio      1.00000000
## train_default_data.interest           0.21684054
## train_default_data.recover            0.04724213
## train_default_data.coll_fee           0.03569578
## train_default_data.out_prncp          0.01971055
## train_default_data.total_cc           0.09722695
## train_default_data.total_acc          -0.07708680
## train_default_data.amount             0.11284018
## train_default_data.monthly_payment     0.12591317
## train_default_data.funded             0.11366017
## train_default_data.total_acc.1        -0.07708680
## train_default_data.term               0.06620962
##                                train_default_data.interest
## train_default_data.credit_ratio      0.21684054
## train_default_data.interest          1.00000000
## train_default_data.recover           0.18256292
## train_default_data.coll_fee          0.11744752
```

```

## train_default_data.out_prncp                -0.03664991
## train_default_data.total_cc                  0.14342975
## train_default_data.total_acc                 -0.02092989
## train_default_data.amount                   0.16880736
## train_default_data.monthly_payment          0.16136454
## train_default_data.funded                   0.16811868
## train_default_data.total_acc.1              -0.02092989
## train_default_data.term                     0.42685737
##
## train_default_data.recover
## train_default_data.credit_ratio              0.047242125
## train_default_data.interest                 0.182562923
## train_default_data.recover                  1.000000000
## train_default_data.coll_fee                 0.854912552
## train_default_data.out_prncp                -0.193158233
## train_default_data.total_cc                 0.120671898
## train_default_data.total_acc                0.002327653
## train_default_data.amount                   0.193499669
## train_default_data.monthly_payment          0.195906686
## train_default_data.funded                   0.193669274
## train_default_data.total_acc.1              0.002327653
## train_default_data.term                     0.106983083
##
## train_default_data.coll_fee
## train_default_data.credit_ratio              0.035695779
## train_default_data.interest                 0.117447521
## train_default_data.recover                  0.854912552
## train_default_data.coll_fee                 1.000000000
## train_default_data.out_prncp                -0.128967058
## train_default_data.total_cc                 0.112718186
## train_default_data.total_acc                0.001107206
## train_default_data.amount                   0.142732734
## train_default_data.monthly_payment          0.149499446
## train_default_data.funded                   0.143175137
## train_default_data.total_acc.1              0.001107206
## train_default_data.term                     0.059918314
##
## train_default_data.out_prncp
## train_default_data.credit_ratio              0.01971055
## train_default_data.interest                 -0.03664991
## train_default_data.recover                  -0.19315823
## train_default_data.coll_fee                 -0.12896706
## train_default_data.out_prncp                1.000000000
## train_default_data.total_cc                 -0.19777498
## train_default_data.total_acc                0.12805758
## train_default_data.amount                   0.53278047
## train_default_data.monthly_payment          0.43759880
## train_default_data.funded                   0.53471542
## train_default_data.total_acc.1              0.12805758
## train_default_data.term                     0.32918838
##
## train_default_data.total_cc
## train_default_data.credit_ratio              0.09722695
## train_default_data.interest                 0.14342975
## train_default_data.recover                  0.12067190
## train_default_data.coll_fee                 0.11271819
## train_default_data.out_prncp                -0.19777498
## train_default_data.total_cc                 1.000000000

```

```

## train_default_data.total_acc          0.12227752
## train_default_data.amount             0.51560491
## train_default_data.monthly_payment    0.54692303
## train_default_data.funded             0.51412522
## train_default_data.total_acc.1        0.12227752
## train_default_data.term                0.08853451
##                                     train_default_data.total_acc
## train_default_data.credit_ratio        -0.077086803
## train_default_data.interest            -0.020929890
## train_default_data.recover             0.002327653
## train_default_data.coll_fee            0.001107206
## train_default_data.out_prncp           0.128057582
## train_default_data.total_cc            0.122277524
## train_default_data.total_acc           1.000000000
## train_default_data.amount              0.217840646
## train_default_data.monthly_payment     0.190361979
## train_default_data.funded              0.218455524
## train_default_data.total_acc.1         1.000000000
## train_default_data.term                0.113760782
##                                     train_default_data.amount
## train_default_data.credit_ratio         0.1128402
## train_default_data.interest             0.1688074
## train_default_data.recover              0.1934997
## train_default_data.coll_fee             0.1427327
## train_default_data.out_prncp            0.5327805
## train_default_data.total_cc             0.5156049
## train_default_data.total_acc            0.2178406
## train_default_data.amount              1.0000000
## train_default_data.monthly_payment      0.9484386
## train_default_data.funded              0.9990489
## train_default_data.total_acc.1          0.2178406
## train_default_data.term                 0.4069540
##                                     train_default_data.monthly_payment
## train_default_data.credit_ratio         0.1259132
## train_default_data.interest             0.1613645
## train_default_data.recover              0.1959067
## train_default_data.coll_fee             0.1494994
## train_default_data.out_prncp            0.4375988
## train_default_data.total_cc             0.5469230
## train_default_data.total_acc            0.1903620
## train_default_data.amount              0.9484386
## train_default_data.monthly_payment      1.0000000
## train_default_data.funded              0.9498231
## train_default_data.total_acc.1          0.1903620
## train_default_data.term                 0.1499891
##                                     train_default_data.funded
## train_default_data.credit_ratio         0.1136602
## train_default_data.interest             0.1681187
## train_default_data.recover              0.1936693
## train_default_data.coll_fee             0.1431751
## train_default_data.out_prncp            0.5347154
## train_default_data.total_cc             0.5141252
## train_default_data.total_acc            0.2184555
## train_default_data.amount              0.9990489

```



```
## train_default_data.monthly_payment      0.9498231
## train_default_data.funded               1.0000000
## train_default_data.total_acc.1          0.2184555
## train_default_data.term                 0.4051272
##                                     train_default_data.total_acc.1
## train_default_data.credit_ratio         -0.077086803
## train_default_data.interest             -0.020929890
## train_default_data.recover              0.002327653
## train_default_data.coll_fee             0.001107206
## train_default_data.out_prncp            0.128057582
## train_default_data.total_cc             0.122277524
## train_default_data.total_acc           1.000000000
## train_default_data.amount              0.217840646
## train_default_data.monthly_payment      0.190361979
## train_default_data.funded               0.218455524
## train_default_data.total_acc.1          1.000000000
## train_default_data.term                 0.113760782
##                                     train_default_data.term
## train_default_data.credit_ratio         0.06620962
## train_default_data.interest             0.42685737
## train_default_data.recover              0.10698308
## train_default_data.coll_fee             0.05991831
## train_default_data.out_prncp            0.32918838
## train_default_data.total_cc             0.08853451
## train_default_data.total_acc           0.11376078
## train_default_data.amount              0.40695402
## train_default_data.monthly_payment      0.14998912
## train_default_data.funded               0.40512719
## train_default_data.total_acc.1          0.11376078
## train_default_data.term                 1.00000000
```

## Using logistic regression

```
suppressWarnings({default_log_model<-glm(default~.,data=train_data, family="binomial")
})
summary(default_log_model)
```

```
##
## Call:
## glm(formula = default ~ ., family = "binomial", data = train_data)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.1793  -0.4475  -0.2404   0.0000   4.2944
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.105e+00  1.223e+00  -4.174 2.99e-05 ***
## n_collect       8.230e-01  5.920e-01   1.390  0.16447
## credit_ratio    6.309e-03  4.520e-03   1.396  0.16279
## interest        2.033e-01  8.717e-02   2.332  0.01972 *
## initial_list_statusb -2.876e-01  2.027e-01  -1.419  0.15599
## recover         1.468e+02  1.234e+02   1.190  0.23412
```

```

## coll_fee      8.990e-01  2.916e+01  0.031  0.97540
## out_prncp    -3.513e-02  1.991e-01 -0.176  0.85996
## total_cc     -1.468e+02  1.234e+02 -1.190  0.23395
## term         1.382e-01  2.537e-01  0.545  0.58582
## fees_rec      1.469e+02  1.234e+02  1.190  0.23388
## total_acc     1.039e-03  1.124e-02  0.092  0.92637
## employment    3.248e-02  3.106e-02  1.045  0.29582
## amount       -4.545e-02  5.267e+01 -0.001  0.99931
## monthly_payment 1.728e-03  2.575e-03  0.671  0.50224
## funded        8.527e-02  5.267e+01  0.002  0.99871
## statuspartial  2.852e-01  2.191e-01  1.302  0.19286
## statusunchecked -1.069e-01  2.717e-01 -0.394  0.69384
## v1           -2.296e-02  1.309e-02 -1.754  0.07949
## int_rec       1.468e+02  1.234e+02  1.190  0.23402
## reasonbusiness 1.204e+00  9.866e-01  1.220  0.22240
## reasoncc      -3.459e-01  6.921e-01 -0.500  0.61723
## reasondebt    -1.566e-01  6.617e-01 -0.237  0.81297
## reasonevent   -2.392e+01  6.688e+05  0.000  0.99997
## reasonholiday -1.927e+01  9.385e+03 -0.002  0.99836
## reasonhome     2.653e+00  1.650e+00  1.608  0.10788
## reasonmedical -7.392e-03  1.270e+00 -0.006  0.99535
## reasonmoving   8.773e-01  1.011e+00  0.868  0.38561
## reasonother    -2.177e-01  7.426e-01 -0.293  0.76940
## reasonrenovation 4.491e-01  7.300e-01  0.615  0.53843
## reasonsolar   -2.340e+01  1.508e+05  0.000  0.99988
## reasontransport -3.503e-01  1.263e+00 -0.277  0.78150
## last_payment  -2.074e-04  7.582e-05 -2.735  0.00624 **
## pymnt_rec     2.369e-02  1.490e-02  1.590  0.11189
## qualityq2     -3.159e-02  5.705e-01 -0.055  0.95583
## qualityq3     -4.528e-01  7.436e-01 -0.609  0.54261
## qualityq4     -3.720e-01  9.546e-01 -0.390  0.69677
## qualityq5     -1.173e+00  1.186e+00 -0.989  0.32258
## qualityq6     -1.275e+00  1.502e+00 -0.849  0.39609
## qualityq7     -1.508e+00  1.769e+00 -0.852  0.39403
## out_prncp_inv -4.722e-03  4.775e-03 -0.989  0.32277
## violations    -1.489e-01  1.831e-01 -0.813  0.41603
## del           7.756e-02  9.040e-02  0.858  0.39092
## inc          -3.517e-06  2.934e-06 -1.199  0.23061
## prin_rec      1.468e+02  1.234e+02  1.190  0.23415
## credit_bal    -1.391e-06  6.229e-06 -0.223  0.82330
## ncc           1.512e-02  2.578e-02  0.586  0.55760
## req           1.222e-01  9.179e-02  1.331  0.18318
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2446.98 on 1919 degrees of freedom
## Residual deviance: 811.31 on 1872 degrees of freedom
## AIC: 907.31
##
## Number of Fisher Scoring iterations: 24

```

```
testProb <- predict(default_log_model, newdata = test_data, type = "response")
# Calculate the error on the test data taken out from train
testActual <- ifelse(testProb>0.5, 1, 0)
error <- sum(abs(testActual - test_data$default)) / nrow(test_data)
error
```

```
## [1] 0.09375
```

```
log_train_loss<-test_data$default*test_data$amount
testLoss <- testProb * test_data$amount

MAE<- sum(abs(log_train_loss-testLoss)) / nrow(test_data)
MAE
```

```
## [1] 2155.379
```

The model is predicting 6.67% incorrect predictions by keeping 0.5 as the threshold using logistic regression.  
The mean absolute error for this training model is \$2014

### Using logistic regression to predict using the actual test data

```
predicted_prob <- predict(default_log_model, newdata = test_default_data, type = "response")
log_predicted_loss=predicted_prob*test_default_data$amount
```

```
log_test_loss<-test_default_data$default*test_default_data$amount

logit_MAE<- sum(abs(log_test_loss -log_predicted_loss)) / nrow(test_default_data)
logit_MAE
```

```
## [1] 2004.124
```

The mean absolute error for this training model is \$1929

### Lasso regression

```
encoded_train_data <- predict(dummyVars("~ .", train_default_data,fullRank = T), newdata = train_default_data)
encoded_test_data <- predict(dummyVars("~ .", test_default_data,fullRank = T), newdata = test_default_data)
encoded_test_data<-encoded_test_data[,-1]

predictors <- encoded_train_data[, -1]
response <- as.matrix(encoded_train_data[, 1])

# perform cross-validation with glmnet
cvfit <- cv.glmnet(encoded_train_data[, -1], encoded_train_data[, 1], alpha = 1, nfolds = 10)

# get the 1SE lambda value
lambda_1se <- cvfit$lambda.1se

# fit the final model with the selected lambda value
lasso_fit <- glmnet(predictors, response, alpha = 1, lambda = lambda_1se)

# # extract the coefficients
coefficients <- coef(lasso_fit)
coefficients
```

```
## 48 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)                1.844500e-01
## n_collect                   .
## credit_ratio                .
## interest                    1.455313e-02
## initial_list_statusb       -2.369468e-02
## recover                     4.720253e-06
## coll_fee                    .
## out_prncp                   -3.079772e-05
## total_cc                    .
## term                        .
## fees_rec                    4.797152e-03
## total_acc                   .
## employment                  .
## amount                      9.655277e-06
## monthly_payment             1.330882e-04
## funded                      1.420111e-05
## statuspartial               .
## statusunchecked              .
## v1                           .
## int_rec                     .
## reasonbusiness              .
## reasoncc                    .
## reasondebt                  .
## reasonevent                 .
## reasonholiday               .
## reasonhome                  .
## reasonmedical               .
## reasonmoving                .
## reasonother                 .
## reasonrenovation            .
## reasonsolar                 .
## reasontransport             .
## last_payment                -1.294515e-05
## pymnt_rec                   .
## qualityq2                   .
## qualityq3                   .
## qualityq4                   .
## qualityq5                   .
## qualityq6                   .
## qualityq7                   .
## out_prncp_inv               -8.300072e-06
## violations                   -1.147107e-02
## del                          .
## inc                          .
## prin_rec                    -3.418576e-05
## credit_bal                  .
## ncc                         .
## req                         3.442775e-03
```

```
lasso_predicted_prob <- predict(lasso_fit, newx= as.matrix(encoded_test_data))
lasso_predicted_loss=lasso_predicted_prob*test_default_data$amount
```

```
lasso_test_loss<-test_default_data$default*test_default_data$amount

lasso_MAE<- sum(abs(lasso_test_loss -lasso_predicted_loss)) / nrow(test_default_data)
lasso_MAE
```

```
## [1] 3591.947
```

Using lasso, 14 coefficients are showing significant and rest all are pushed to zero. MAE is coming out as 3579

## PCA

```
scaled_train_data <- scale(encoded_train_data)
scaled_test_data <- scale(encoded_test_data)
scaled_train_data<-scaled_train_data[, -1]

pca <- prcomp(scaled_train_data)
summary(pca)
```

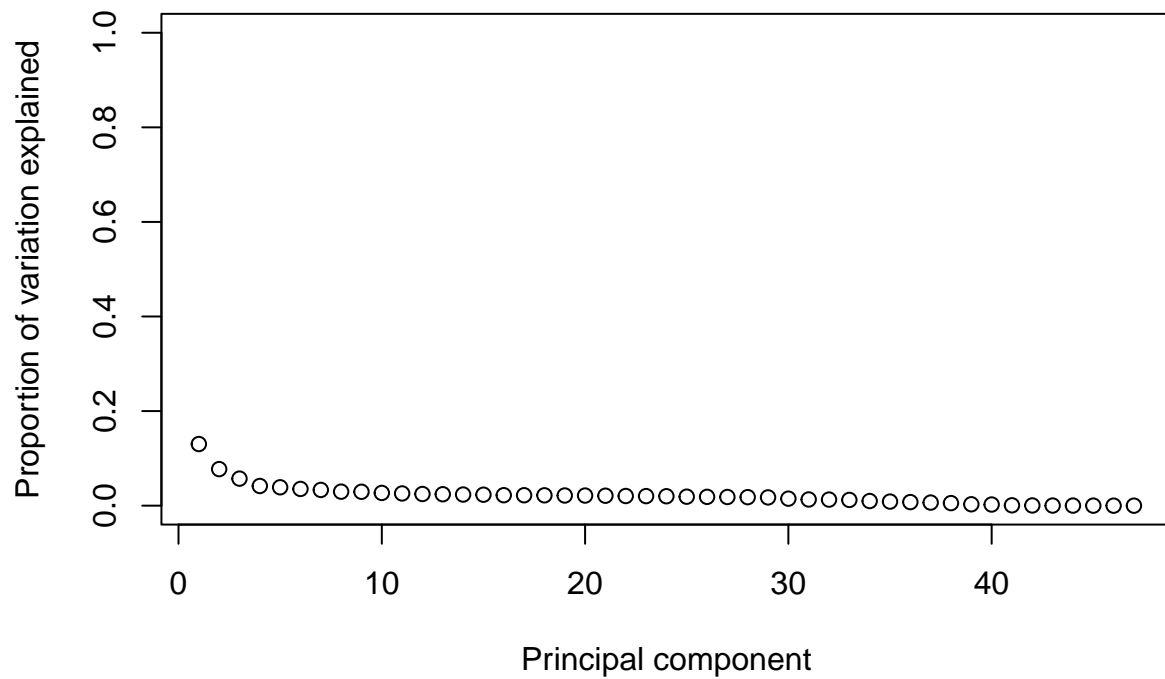
```
## Importance of components:
```

```
##              PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation  2.4748 1.90585 1.63922 1.39851 1.35061 1.28917 1.25007
## Proportion of Variance 0.1303 0.07728 0.05717 0.04161 0.03881 0.03536 0.03325
## Cumulative Proportion 0.1303 0.20760 0.26477 0.30638 0.34519 0.38056 0.41380
##              PC8      PC9      PC10     PC11     PC12     PC13     PC14
## Standard deviation  1.17914 1.17465 1.12470 1.10084 1.07630 1.06408 1.05028
## Proportion of Variance 0.02958 0.02936 0.02691 0.02578 0.02465 0.02409 0.02347
## Cumulative Proportion 0.44339 0.47274 0.49966 0.52544 0.55009 0.57418 0.59765
##              PC15     PC16     PC17     PC18     PC19     PC20     PC21
## Standard deviation  1.04388 1.01901 1.01760 1.01365 1.00508 1.00158 0.99391
## Proportion of Variance 0.02318 0.02209 0.02203 0.02186 0.02149 0.02134 0.02102
## Cumulative Proportion 0.62083 0.64293 0.66496 0.68682 0.70831 0.72966 0.75068
##              PC22     PC23     PC24     PC25     PC26     PC27     PC28
## Standard deviation  0.9792 0.97395 0.96618 0.94399 0.93593 0.92712 0.91944
## Proportion of Variance 0.0204 0.02018 0.01986 0.01896 0.01864 0.01829 0.01799
## Cumulative Proportion 0.7711 0.79126 0.81112 0.83008 0.84872 0.86701 0.88499
##              PC29     PC30     PC31     PC32     PC33     PC34     PC35
## Standard deviation  0.90591 0.83424 0.78377 0.77678 0.75490 0.67809 0.63619
## Proportion of Variance 0.01746 0.01481 0.01307 0.01284 0.01212 0.00978 0.00861
## Cumulative Proportion 0.90245 0.91726 0.93033 0.94317 0.95529 0.96508 0.97369
##              PC36     PC37     PC38     PC39     PC40     PC41     PC42
## Standard deviation  0.59604 0.55301 0.50241 0.37132 0.33635 0.18654 0.14051
## Proportion of Variance 0.00756 0.00651 0.00537 0.00293 0.00241 0.00074 0.00042
## Cumulative Proportion 0.98125 0.98775 0.99312 0.99606 0.99846 0.99921 0.99963
##              PC43     PC44     PC45     PC46     PC47
## Standard deviation  0.10214 0.07937 0.02966 0.001502 9.084e-08
## Proportion of Variance 0.00022 0.00013 0.00002 0.000000 0.000e+00
## Cumulative Proportion 0.99985 0.99998 1.00000 1.000000 1.000e+00
```

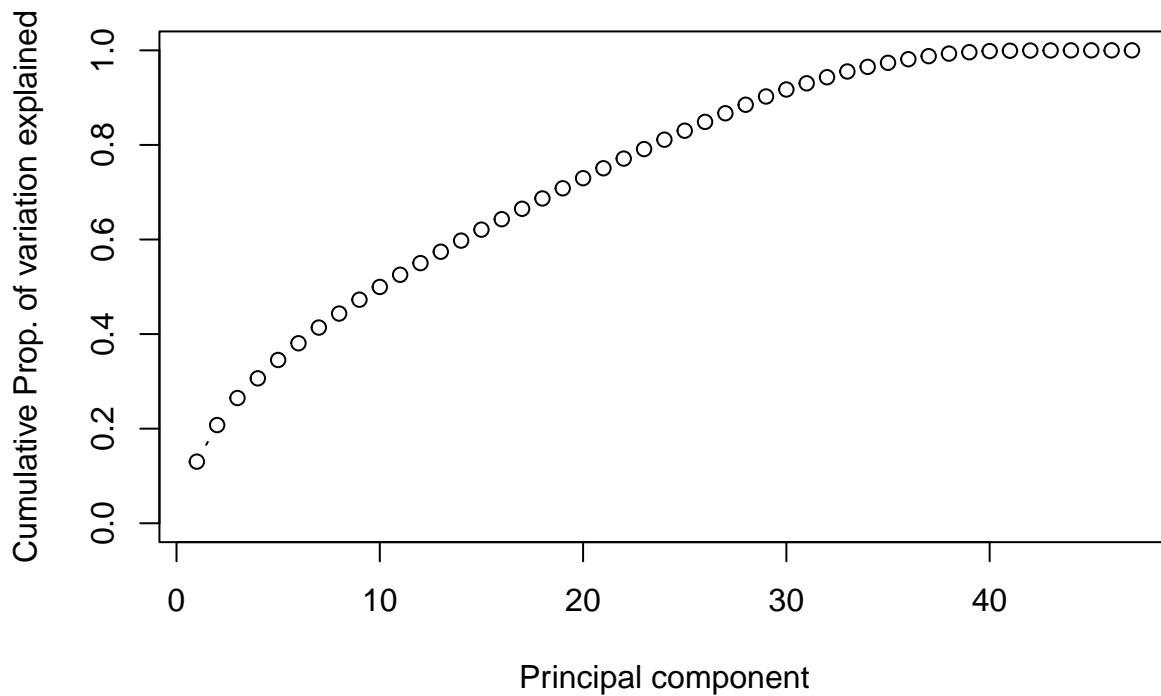
```
pca.var <- pca$sdev^2

pve <- pca.var/sum(pca.var)
```

```
plot(pve, xlab = "Principal component",
     ylab = "Proportion of variation explained",
     ylim = c(0, 1),
     type = 'b')
```



```
plot(cumsum(pve), xlab = "Principal component",
     ylab = "Cumulative Prop. of variation explained",
     ylim = c(0, 1),
     type = 'b')
```



*#Based on the summary function and the elbow curve, picking top 24 principal components out of 48 that*

```
pca_data<-data.frame(Default=encoded_train_data[, 'default'],pca$x[,1:24])
```

*# Train a logistic regression model on the transformed data*

```
pca_logit_model <- glm(pca_data$Default ~ ., data = pca_data, family = "binomial")
```

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

```
test.p <- predict(pca, newdata = encoded_test_data[,])
```

*# Make predictions on the testing data*

```
test_pca <- predict(pca_logit_model, newdata=as.data.frame(test.p),type="response")
```

*# Evaluate the performance of the logistic regression model*

```
pca_prediction<-ifelse(test_pca > 0.5, 1,0)
```

*#Below table summarizes the true positives and false positives prediction*

```
table(test_default_data$default, pca_prediction)
```

```
##      pca_prediction
##      0      1
## 0 400      2
## 1  49    149
```

```
pca_predicted_loss=test_pca*test_default_data$amount
pca_test_loss<-test_default_data$default*test_default_data$amount

pca_MAE<- sum(abs(pca_predicted_loss -pca_test_loss)) / nrow(test_default_data)
pca_MAE
```

```
## [1] 1344.25
```

The mean absolute error is 1344 for variables selected through PCA.

## PLS

```
# Fit the PLS model with M chosen by cross-validation
pls_default_fit <- train(as.factor(default)~.,data=encoded_train_data, method = "pls",
                        tuneLength = 10, trControl = trainControl(method = "cv", number = 10),
                        preProcess = c("center", "scale"))
pls_m <- pls_default_fit$bestTune$ncomp
pls_m
```

```
## [1] 9
```

```
# Fit the final PLS model with the selected M
pls_model <- plsrf(default ~ ., data = as.data.frame(encoded_train_data), ncomp = pls_m)
pls_prob <- predict(pls_model, newdata = encoded_test_data)

pls_prediction<-ifelse(pls_prob > 0.5, 1,0)

pls_predicted_loss=pls_prob*test_default_data$amount
pls_test_loss<-test_default_data$default*test_default_data$amount

pls_MAE<- sum(abs(pls_predicted_loss -pls_test_loss)) / nrow(test_default_data)
pls_MAE
```

```
## [1] 37871.34
```

The loss for PLS is sky rocketing with 41528

## Weighted logistic

```
sum(train_default_data$default==0)
```

```
## [1] 1598
```

```
sum(train_default_data$default==1)
```

```
## [1] 802
```



```
w1=1
w2=50
weight <- ifelse(train_default_data$default==0, 50, 1)
```

```
suppressWarnings({weighted_log_model<-glm(default~.,data=train_default_data, family="binomial",weights =
)})
summary(weighted_log_model)
```

```
##
## Call:
## glm(formula = default ~ ., family = "binomial", data = train_default_data,
##      weights = weight)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.6426  -0.4463  -0.2587   0.0000   5.4557
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -9.635e+00  1.029e+00  -9.360  < 2e-16 ***
## n_collect       4.849e-01  4.904e-01   0.989  0.322778
## credit_ratio    5.320e-03  3.627e-03   1.467  0.142486
## interest        1.920e-01  6.669e-02   2.880  0.003982 **
## initial_list_statusb -1.026e-01  1.632e-01  -0.629  0.529571
## recover         1.867e+02  1.001e+02   1.865  0.062160 .
## coll_fee        2.337e-01  3.233e+01   0.007  0.994233
## out_prncp       -3.877e-02  4.636e-01  -0.084  0.933355
## total_cc        -1.867e+02  9.982e+01  -1.870  0.061503 .
## term            1.613e-01  2.007e-01   0.803  0.421717
## fees_rec        1.867e+02  9.982e+01   1.870  0.061462 .
## total_acc       6.672e-03  8.841e-03   0.755  0.450459
## employment      1.394e-02  2.447e-02   0.569  0.569019
## amount          -2.642e-03  4.768e+01   0.000  0.999956
## monthly_payment  2.687e-03  2.014e-03   1.334  0.182078
## funded          4.200e-02  4.768e+01   0.001  0.999297
## statuspartial    5.440e-01  1.813e-01   3.001  0.002689 **
## statusunchecked  2.931e-01  2.245e-01   1.305  0.191832
## v1              -1.237e-02  1.043e-02  -1.187  0.235340
## int_rec         1.866e+02  9.982e+01   1.870  0.061518 .
## reasonbusiness   1.124e+00  7.894e-01   1.423  0.154610
## reasoncc         -1.400e-01  6.246e-01  -0.224  0.822605
## reasondebt       2.931e-01  5.956e-01   0.492  0.622647
## reasonevent      -1.844e+01  4.778e+04   0.000  0.999692
## reasonholiday    -1.791e+01  4.260e+03  -0.004  0.996645
## reasonhome       4.214e+00  1.065e+00   3.956  7.63e-05 ***
## reasonmedical    9.271e-01  9.358e-01   0.991  0.321828
## reasonmoving     4.434e-01  8.884e-01   0.499  0.617709
## reasonother      2.621e-01  6.517e-01   0.402  0.687600
## reasonrenovation  7.487e-01  6.405e-01   1.169  0.242423
## reasonsolar      -2.200e+01  9.923e+04   0.000  0.999823
## reasontransport  1.519e-01  1.168e+00   0.130  0.896545
## last_payment     -2.614e-04  7.561e-05  -3.457  0.000545 ***
```

```
## pymnt_rec          1.182e-02  7.844e-03   1.507 0.131784
## qualityq2         -7.671e-02  4.641e-01  -0.165 0.868708
## qualityq3         -5.914e-01  5.919e-01  -0.999 0.317663
## qualityq4         -5.043e-01  7.547e-01  -0.668 0.503974
## qualityq5         -9.470e-01  9.074e-01  -1.044 0.296664
## qualityq6         -9.369e-01  1.134e+00  -0.826 0.408760
## qualityq7         -1.567e+00  1.378e+00  -1.137 0.255498
## out_prncp_inv     -6.484e-04  3.640e-03  -0.178 0.858618
## violations        -1.254e-01  1.455e-01  -0.862 0.388780
## del               4.261e-02  7.229e-02   0.589 0.555541
## inc              -2.022e-06  2.334e-06  -0.866 0.386332
## prin_rec          1.866e+02  9.982e+01   1.869 0.061576 .
## credit_bal       -5.489e-06  6.034e-06  -0.910 0.363050
## ncc              -1.885e-03  2.020e-02  -0.093 0.925640
## req               1.867e-01  7.229e-02   2.582 0.009819 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 8992.7  on 2399  degrees of freedom
## Residual deviance: 2469.7  on 2352  degrees of freedom
## AIC: 2565.7
##
## Number of Fisher Scoring iterations: 24
```

```
testProb <- predict(weighted_log_model, newdata = test_default_data, type = "response")
# Calculate the error on the test data taken out from train
testActual <- ifelse(testProb>0.5, 1, 0)
error <- sum(abs(testActual - test_default_data$default)) / nrow(test_default_data)
error
```

```
## [1] 0.07166667
```

```
weighted_train_loss<-test_default_data$default*test_default_data$amount
weightedLoss <- testProb * test_default_data$amount

w_MAE<- sum(abs(weighted_train_loss-weightedLoss)) / nrow(test_default_data)
w_MAE
```

```
## [1] 1226.233
```

The data is imbalanced with approx 50% more instances of 0 than 1 in the default column. Hence applied weighted logistic regression. I have give weight of 50 to “0” and 1 to “1” in the regression. The error is 0.07 and the MAE is 1226. This is the least MAE.

**Model Selection Steps** We first started with pre-processing data. Some of the steps involved in pre-processing are: 1. Converting numerical variables to correct format 2. Stripping away characters from ‘term’ column to make it suitable for use in regression 3. Checking for NAs 4. Replacing NAs with mean value of columns based on the frequency of occurrence 5. Converting categorical variables to dummy

Then, we also looked at the scatter plots of all the numerical variables to find if there is a need of variable transformation. All the plots showed random pattern.

The first model I tried is logistic regression as this is a clear classification problem. I divided the training data further into train and test for this method. Then, I calculated the MAE for logistic using the test data from training set as well as the actual test set. The MAE for actual test set is 1929

Then, I moved on to check for lasso regression. There were 14 significant variables and the MAE value was 3579.

The next model I tried is logistic but using Principal component analysis. PCA is a good approach to apply for dimensionality reduction. Since, I didn't find a good number of significant variables through lasso, PCA seemed to be the next best approach. And after fitting PCA and using actual test data, MAE was 1344.

Although I also tried to fit a PLS model but it performed bad because these are best for continuous variables. PLS assumes a linear relationship between the independent and dependent variables. While this assumption is reasonable for many regression problems, it may not hold for classification problems, where the relationship between the independent and dependent variables may be more complex and nonlinear.

The next model was weighted logistic regression. The data is imbalanced with approx 50% more instances of 0 than 1 in the default column. Hence applied weighted logistic regression. I have give weight of 50 to "0" and 1 to "1" in the regression. The error is 0.07 and the MAE is 1226. This is the least MAE. And hence this is the final model.

This model has the least mean absolute error and is a good fit for this imbalanced dataset.