# 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_

#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_da
```

### 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))
```

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

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

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

```r
# 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)
```

```r
#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$reco

# pairs(cor_data, pch = 19)

#Correlation
# cor(cor_data)
```

### Using logistic regression

```r
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
##  -8.49    0.00    0.00    0.00    8.49
##
## Coefficients:
##                        Estimate Std. Error    z value Pr(>|z|)
## (Intercept)          -1.155e+15  2.050e+07  -56328307   <2e-16 ***
## n_collect            -1.663e+14  1.441e+07  -11540679   <2e-16 ***
## credit_ratio          2.271e+12  7.298e+04   31113555   <2e-16 ***
## interest              5.966e+13  1.305e+06   45721524   <2e-16 ***
```

```
## initial_list_statusb -1.285e+14  3.386e+06  -37956687   <2e-16 ***
## recover               2.850e+16  1.648e+09   17295629   <2e-16 ***
## coll_fee              1.132e+12  1.809e+04   62612591   <2e-16 ***
## out_prncp            -4.571e+12  9.282e+04  -49249189   <2e-16 ***
## total_cc             -2.850e+16  1.648e+09  -17295854   <2e-16 ***
## term                  1.448e+14  4.708e+06   30759610   <2e-16 ***
## fees_rec              2.851e+16  1.648e+09   17300761   <2e-16 ***
## total_acc             5.892e+12  1.907e+05   30889209   <2e-16 ***
## employment           -5.552e+12  5.118e+05  -10848739   <2e-16 ***
## amount               -9.730e+10  4.534e+03  -21462671   <2e-16 ***
## monthly_payment       2.173e+12  4.647e+04   46758004   <2e-16 ***
## funded                2.696e+11  4.793e+03   56252173   <2e-16 ***
## statuspartial         1.004e+14  3.793e+06   26463447   <2e-16 ***
## statusunchecked       2.740e+13  4.136e+06    6625988   <2e-16 ***
## v1                   -1.004e+13  2.234e+05  -44944094   <2e-16 ***
## int_rec               2.850e+16  1.648e+09   17295858   <2e-16 ***
## reasonbusiness        5.380e+14  1.615e+07   33303772   <2e-16 ***
## reasoncc              1.636e+14  1.159e+07   14120049   <2e-16 ***
## reasondebt            2.564e+14  1.122e+07   22861449   <2e-16 ***
## reasonevent           3.101e+14  4.058e+07    7642095   <2e-16 ***
## reasonholiday        -3.375e+14  2.178e+07  -15496750   <2e-16 ***
## reasonhome            6.048e+14  2.545e+07   23762902   <2e-16 ***
## reasonmedical         9.113e+14  1.885e+07   48345258   <2e-16 ***
## reasonmoving          1.546e+14  1.776e+07    8705223   <2e-16 ***
## reasonother           3.142e+12  1.278e+07     245868   <2e-16 ***
## reasonrenovation      3.472e+14  1.262e+07   27511561   <2e-16 ***
## reasonsolar          -1.767e+15  4.901e+07  -36041851   <2e-16 ***
## reasontransport       6.000e+14  2.019e+07   29720431   <2e-16 ***
## last_payment         -2.131e+11  6.394e+02 -333235149   <2e-16 ***
## pymnt_rec             6.224e+10  2.036e+03   30575310   <2e-16 ***
## qualityq2            -2.255e+14  7.121e+06  -31669197   <2e-16 ***
## qualityq3            -3.433e+14  1.014e+07  -33872196   <2e-16 ***
## qualityq4            -3.561e+14  1.368e+07  -26029881   <2e-16 ***
## qualityq5            -5.917e+14  1.735e+07  -34092374   <2e-16 ***
## qualityq6            -6.279e+14  2.227e+07  -28198102   <2e-16 ***
## qualityq7            -1.362e+15  2.676e+07  -50885398   <2e-16 ***
## out_prncp_inv         4.335e+12  9.286e+04   46680546   <2e-16 ***
## violations           -1.254e+14  2.429e+06  -51601303   <2e-16 ***
## del                   4.125e+13  1.695e+06   24341038   <2e-16 ***
## inc                  -2.046e+09  4.329e+01  -47261818   <2e-16 ***
## prin_rec              2.850e+16  1.648e+09   17295646   <2e-16 ***
## credit_bal           -5.655e+09  9.415e+01  -60065470   <2e-16 ***
## ncc                   5.146e+12  4.332e+05   11879885   <2e-16 ***
## req                   2.170e+13  1.456e+06   14901000   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  2440  on 1919  degrees of freedom
## Residual deviance: 22708  on 1872  degrees of freedom
## AIC: 22804
##
## Number of Fisher Scoring iterations: 25
```

```r
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] 0.175
```

```r
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] 2399.583
```

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**

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

```r
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] 1980.25
```

The mean absolute error for this training model is $1929

**Lasso regression**

```r
encoded_train_data <- predict(dummyVars("~ .", train_default_data,fullRank = T), newdata = train_defaul
encoded_test_data <- predict(dummyVars("~ .", test_default_data,fullRank = T), newdata = test_default_da
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)           1.840012e-01
## n_collect             .
## credit_ratio          .
## interest              1.430080e-02
## initial_list_statusb -2.791325e-02
## recover               .
## coll_fee              .
## out_prncp            -3.319851e-05
## total_cc              .
## term                  .
## fees_rec              5.069290e-03
## total_acc             .
## employment            .
## amount                9.109910e-06
## monthly_payment       1.309849e-04
## funded                1.689420e-05
## statuspartial         .
## statusunchecked       .
## v1                    .
## int_rec               .
## reasonbusiness        .
## reasoncc              .
## reasondebt            .
## reasonevent           .
## reasonholiday         .
## reasonhome            2.960138e-02
## reasonmedical         .
## reasonmoving          .
## reasonother           .
## reasonrenovation      .
## reasonsolar           .
## reasontransport       .
## last_payment         -1.299435e-05
## pymnt_rec             .
## qualityq2             .
## qualityq3             .
## qualityq4             .
## qualityq5             .
## qualityq6             .
## qualityq7             .
## out_prncp_inv        -7.785054e-06
## violations           -1.489845e-02
## del                   .
## inc                   .
## prin_rec             -3.649823e-05
## credit_bal            .
## ncc                   .
## req                   5.113027e-03
```

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

```r
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] 3572.071
```

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

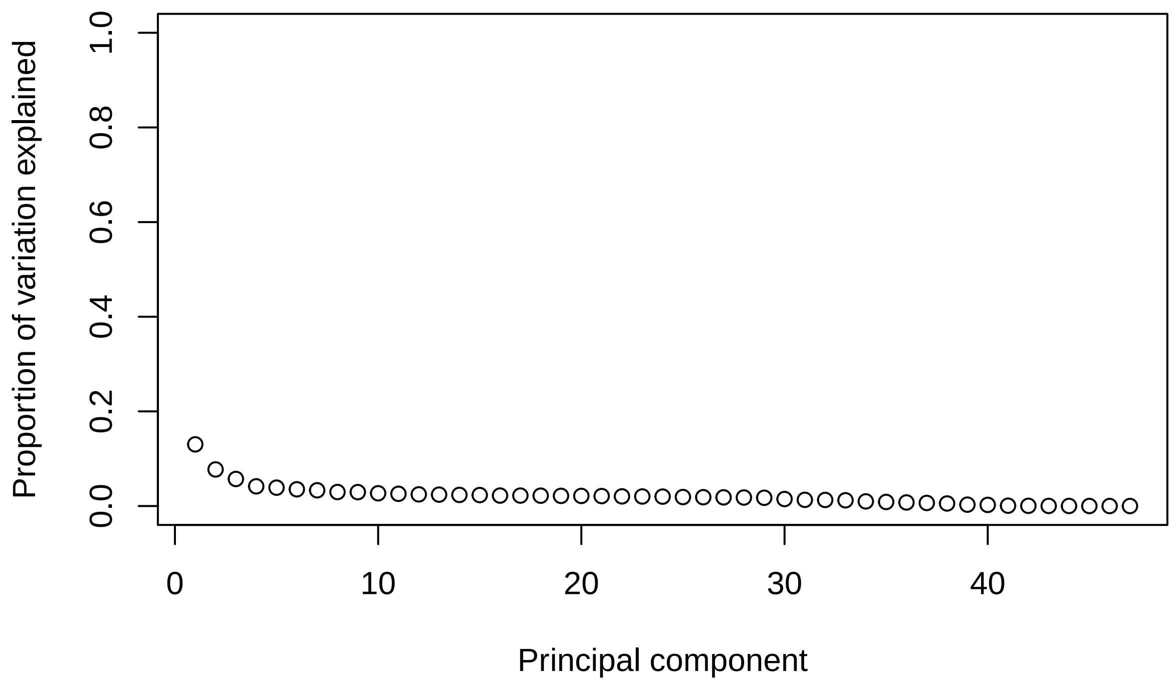
**PCA**

```r
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)
```

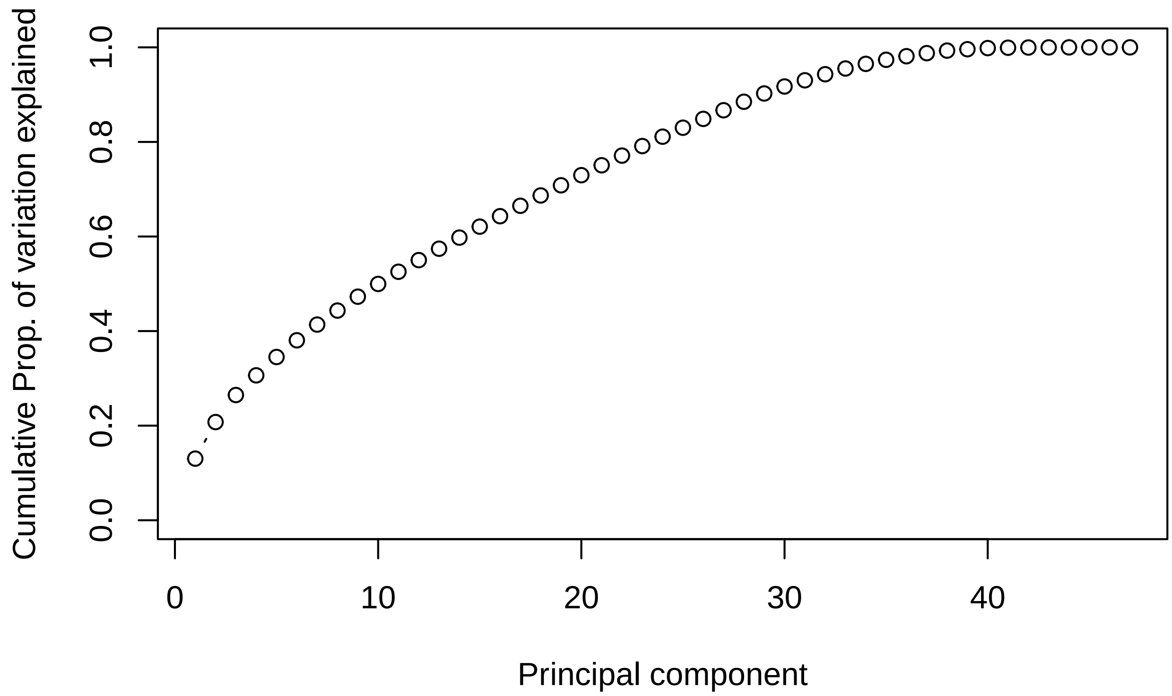
```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4748 1.90585 1.63922 1.39851 1.35061 1.28917 1.25007
## Proportion of Variance 0.1303 0.07728 0.05717 0.04161 0.03881 0.03536 0.03325
## Cumulative Proportion  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
```

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

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

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




```r
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
```

```r
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**

```r
# 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] 10
```

```r
# Fit the final PLS model with the selected M
pls_model <- plsr(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] 41528.23
```

The loss for PLS is sky rocketing with 41528

**Weighted logistic**

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

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

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

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

```r
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
```

```r
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
```

```r
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 occurence 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 datset.