SM_HW3

Kwon Namtaek

2020 11 22

과목 : 통계적 모델링과 머신러닝

통계학과 2015313693 권남택

```
library(mice)
library(rms)
library(finalfit)
library(caret) # create CV
library(car)
library(bestNormalize) # yeo-johnson transformation
library(gridExtra) # Visualization
library(glmnet)
library(data.table) # data loading
library(tidyverse) # data preprocess
```

필요한 라이브러리 로드

```
train = fread('train.csv', data.table = F)
train %>% dim # 2000 행 25 열

## [1] 2000   25

train = train %>% select(-V1) # 인덱스이므로 삭제,
```

범주형 변수들을 factor 처리 해주었는데, 해당 코드는 숨김처리해두었다.

변수 이름을 알기 편하게 설정했다.

```
colnames(train) = c('credit', 'gender', 'edu', 'martial', 'age',
                    'april_pay_rec', 'may_pay_rec', 'june_pay_rec', 'july_pay_rec', 'aug_pay_rec', 'sep_pay_rec',
                    'april_state', 'may_state', 'june_state', 'july_state', 'aug_state', 'sep_state',
                    'april_pay_pre', 'may_pay_pre', 'june_pay_pre', 'july_pay_pre', 'aug_pay_pre', 'sep_pay_pre',
                    'default')
train_colnames = colnames(train)
```

데이터를 살펴보고 다음과 같은 내용을 전처리와 모델링 이전에 고려하였다.

1. 결측치가 존재한다. 결측치의 발생 형태를 파악하고, 대체 여부를 결정한다.

2. x3 x4 에 원래 데이터에 있지 않을 범주가 있다(education level 이 56, martial level 이 0). train 과 test 에 동시에 있으므로 이를 적절하게 처리해줘야한다.

3. x5, x6~x11 변수의 범주가 너무 많다. 나이대 그룹핑, 월별 연체여부 -> 연체횟수로 차원을 줄일 수 있지 않을까?

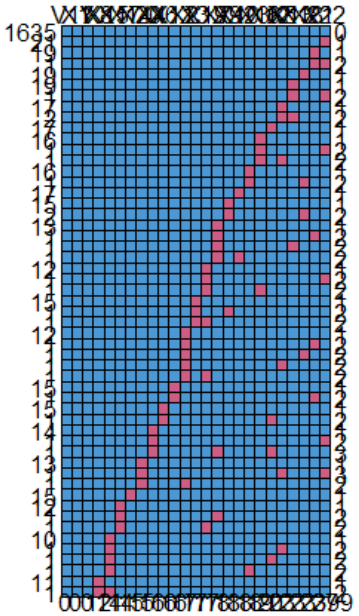
4. 전체 변수가 너무 많다. 월별 previous payment 나 bill statement 의 경우 차원축소가 가능할것 같다.

5. 클래스 불균형은 없다.

EDA & Preprocess 1. 결측치의 처리

다른 전처리를 거치기 이전에, 무엇보다 먼저 결측치 처리를 먼저 진행해야 한다. Imputation 을 거치기 이전에, 이런 결측치가 어떤 형태로 발생했는지, MCAR, MAR, MNAR 중 어느 형태인지 이해해야한다. 먼저 결측치는 전체 관측치 2000 개중에 약 1%에 그친다. Y 와 X1(credit)을 제외한 다른 변수들에 결측치가 각각 약 20 개 정도 발생했는데, 이를 확인하자.

```
md.pattern(train)
```



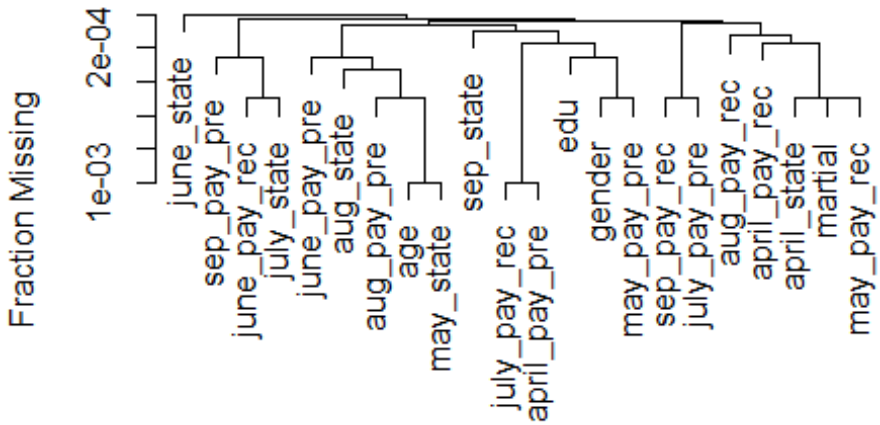
그림을 보면 알 수 있듯, 꼭 어떤 변수로 인해 결측치가 발생한다고 말하기 어렵다. 예를 들어 체납한 이후의 기록이 없는 형태와 같이 변수들 간의 관계로 결측치가 발생하는 것이 아니라고 추측할 수 있다.

또한 결측치가 상당히 적기 때문에, 결측치가 발생하는 구조를 이해하기 어려운데, 이후의 결과들도 유사하다.

```
md.pairs(train)$mm
```

결과값은 불필요하게 페이지를 늘리므로 뺐지만, 매우 sparse 한 형태의 행렬로, 패턴을 관측하기 어렵다.

```
missing.clus = naclus(train %>% select(-credit, -default), method = 'average')
plot(missing.clus)
```



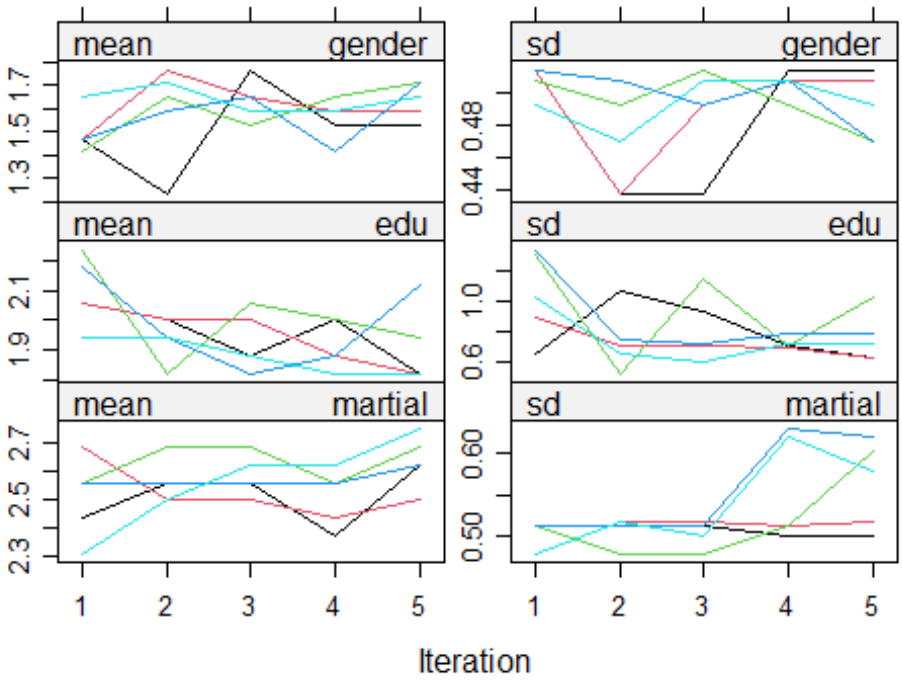
명확하게 군집화할 수 있는 형태가 아니다.

결론적으로 우리의 결측치는 어떤 형태인가? 먼저 명백하게 MNAR 은 아니다. 그렇다면 MCAR 혹은 MAR 로 볼 수 있는데, 비록 MCAR 로 여겨질 수 있더라도, 이는 결측치가 적어서 파악하지 못했기 때문일 수 있다. 따라서 **최대한 많은 관측치를 사용해 정보의 손실을 최소화한 모델링**을 진행할 수 있도록, MICE 방법을 통해 다중대체를 진행하겠다.

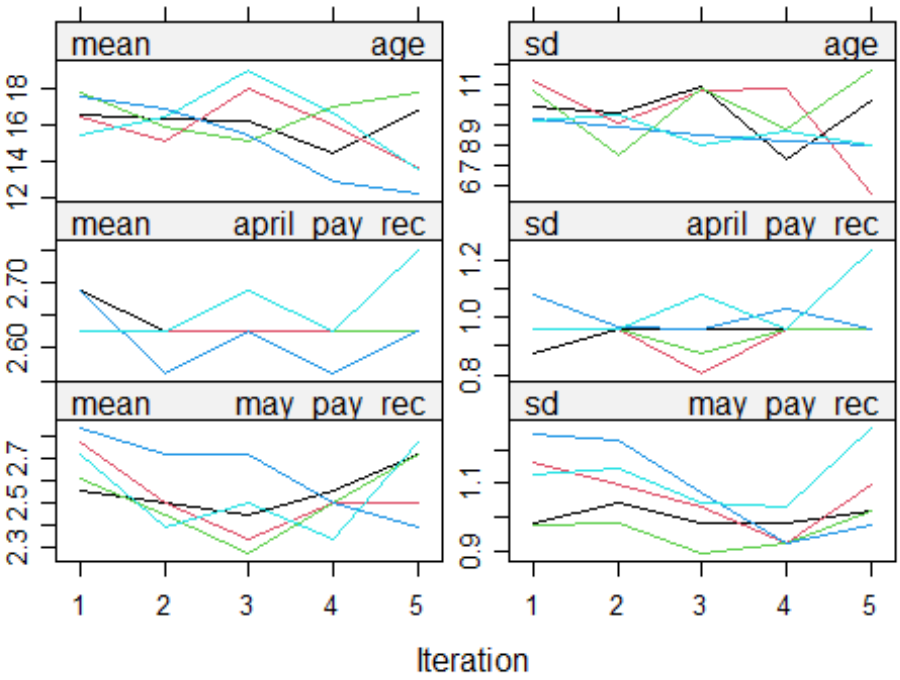
이때 변수들의 결측치를 예측하는 모델로는 CART 를 사용했다. 많은 변수들을 대체해야 하기 때문에, 계산량이 많은 방법은 사용하지 않으려 했다. 로지스틱 계열 모형의 경우, 수렴하지 않는 경우가 존재했고, PMM 또한 singular 문제가 발생했다. 따라서 빠르면서 수렴문제가 비교적 적은 트리 모형을 사용했다. 또한 결측치가 없는 경우, impute 를 지정해주지 않았고, Y(default)의 경우 mice 과정에서 제외해서 정보 유출을 막았다.

```
impute_vec = c('', rep('cart', 22), '')

imputation_pred_mat = matrix(1, ncol = length(train_colnames), nrow = length(train_colnames))
rownames(imputation_pred_mat) = colnames(imputation_pred_mat) = train_colnames
diag(imputation_pred_mat) = 0
```

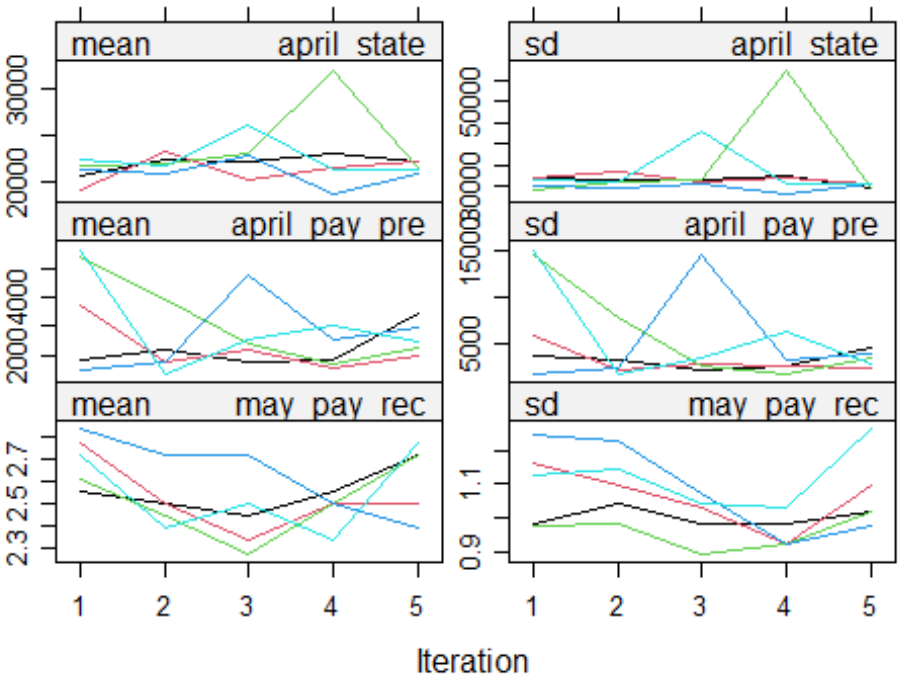
```
imputation_pred_mat[, 'default'] = 0

set.seed(42)
mult_imp = mice(train, m = 5, method = impute_vec, predictorMatrix = imputation_pred_mat, print =
F, seed = 42)

## Warning: Number of logged events: 550

plot(mult_imp, c('gender', 'edu', 'martial'))
```




```
plot(mult_imp, c('age', 'april_pay_rec', 'may_pay_rec'))
```




```
plot(mult_imp, c('april_state', 'april_pay_pre', 'may_pay_rec'))
```

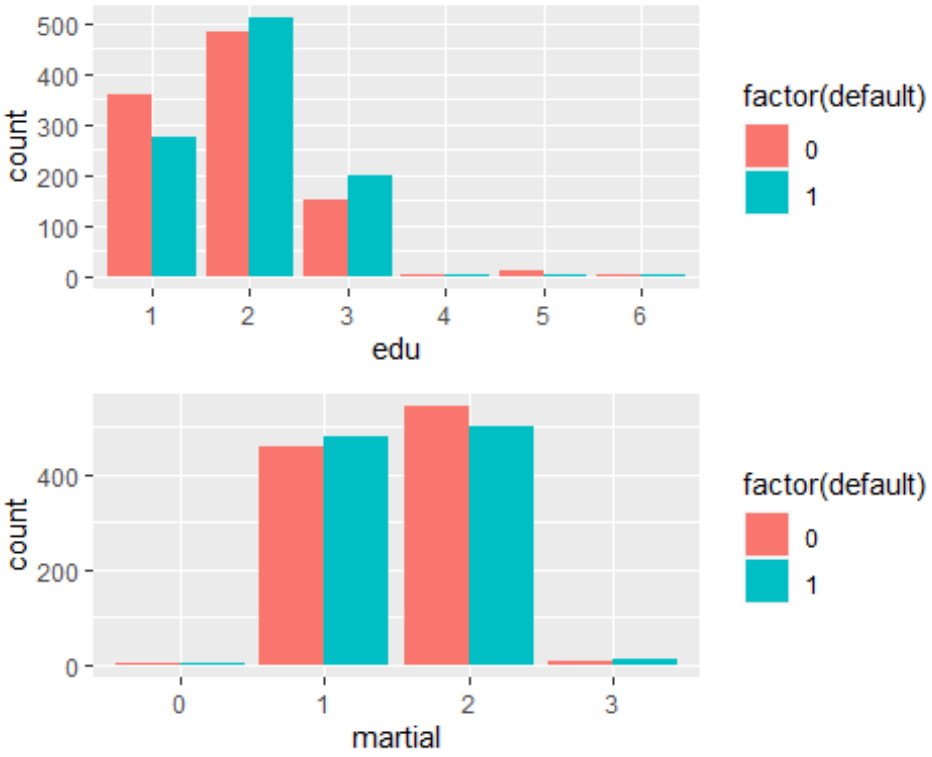


일부만 보였지만, 수렴하는 형태가 전체적으로 나타났다. 이후 density plot 도 확인했는데, july state 의 경우는 분포를 잘 따라가지 못했지만, 나머지 경우는 모두 안정적으로 학습해 결측치를 대체했다.

```
train_imputed = mult_imp %>% complete
```

EDA & Preprocess 2. 재범주화

```
plot_edu = train_imputed %>% ggplot(aes(x = edu, fill = factor(default))) + geom_bar(position = 'dodge')
plot_martial = train_imputed %>% ggplot(aes(x = martial, fill = factor(default))) + geom_bar(position = 'dodge')
grid.arrange(plot_edu, plot_martial, nrow = 2)
```



education 과 martial 변수를 보면 현재 기존 변수 구성에서 없는 변수가 존재한다. 이들을 모두 other 로 넘겨준다. default(Y)나 credit 과 같이 볼 때 명확한 패턴이 없으므로 기타로 처리한다. 이들을 결측치로 처리할 경우, test set 도 동일하게 다루는 것이 매우 어려워지기 때문에 분석의 흐름을 위해 해당 방법으로 처리한다.

```
train_imputed$edu[which(train_imputed$edu %in% c(5, 6))] = 4
train_imputed$edu = train_imputed$edu %>% factor
train_imputed$martial[which(train_imputed$martial == 0)] = 3
train_imputed$martial = train_imputed$martial %>% factor
```

또한 현재 monthly payment record 와 관련한 범주형 변수들이 너무 많다. 이를 통합하겠다. 해당 변수들의 음수 값들을 미리 납입한 경우를 말한다. 여기서 중요한것은 **연체했느냐, 아니냐** 여부이므로, 0 을 기준으로 연체여부 변수로 변환한다. 이후 총 연체횟수라는 범주형 변수를 만들어준다. 이렇게 범주들을 재범주화함으로써, 차원을 크게 감소시킬 수 있으며, 유의미한 정보만 사용할 수 있다.

```
train_imputed$april_pay_rec = ifelse(as.numeric(as.character(train_imputed$april_pay_rec)) < 0.1, 0, 1)
train_imputed$may_pay_rec = ifelse(as.numeric(as.character(train_imputed$may_pay_rec)) < 0.1, 0, 1)
train_imputed$june_pay_rec = ifelse(as.numeric(as.character(train_imputed$june_pay_rec)) < 0.1, 0, 1)
train_imputed$july_pay_rec = ifelse(as.numeric(as.character(train_imputed$july_pay_rec)) < 0.1, 0, 1)
train_imputed$aug_pay_rec = ifelse(as.numeric(as.character(train_imputed$aug_pay_rec)) < 0.1, 0, 1)
train_imputed$sep_pay_rec = ifelse(as.numeric(as.character(train_imputed$sep_pay_rec)) < 0.1, 0, 1)

train_imputed$num_late = train_imputed$april_pay_rec + train_imputed$may_pay_rec +
  train_imputed$june_pay_rec + train_imputed$july_pay_rec +
  train_imputed$aug_pay_rec + train_imputed$sep_pay_rec
train_imputed$num_late = factor(train_imputed$num_late)

train_imputed = train_imputed %>% select(-april_pay_rec, -may_pay_rec, -june_pay_rec, -july_pay_re
c, -aug_pay_rec, -sep_pay_rec)
```

또한 age 변수의 범주를 줄여준다. 연령대별로 재범주화함으로써 **연령대 효과**를 잡아내고 싶으며, 차원또한 감소시킬 수 있다.

```
train_imputed$age = train_imputed$age %>% as.character() %>% as.numeric()
train_imputed$age = ifelse(train_imputed$age < 29.5, '20',
                           ifelse(train_imputed$age < 39.5, '30',
                                  ifelse(train_imputed$age < 49.5, '40',
                                         ifelse(train_imputed$age < 59.5, '50', 'over 60')))) %>% as.factor
```

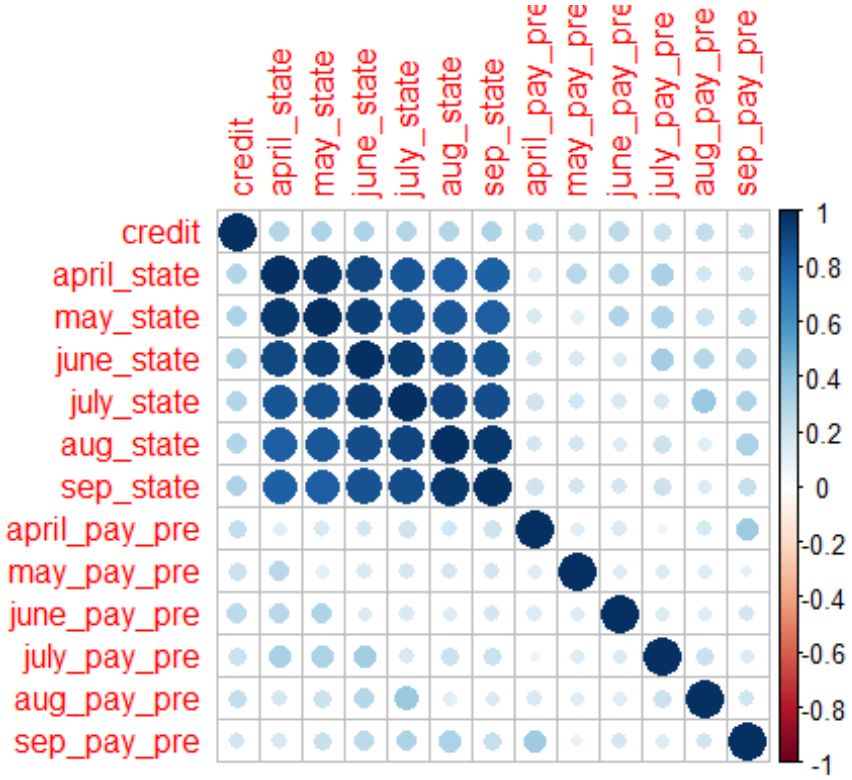
EDA & Preprocess 3. x 변수 변환

x 변수들의 경우 skewness 가 매우 심한 경우들이 많다. 이후 PCA 나 x 변수들의 고차항을 고려할 때를 위해서 Transformation 을 고려해주어야 한다. 이때 음의 값을 갖는 경우도 있으므로, 전체적으로 적용할 수 있도록 Yeo-Johnson Transformation 을 시행한다. 코드는 생략한다.

EDA & Preprocess 4. 차원축소

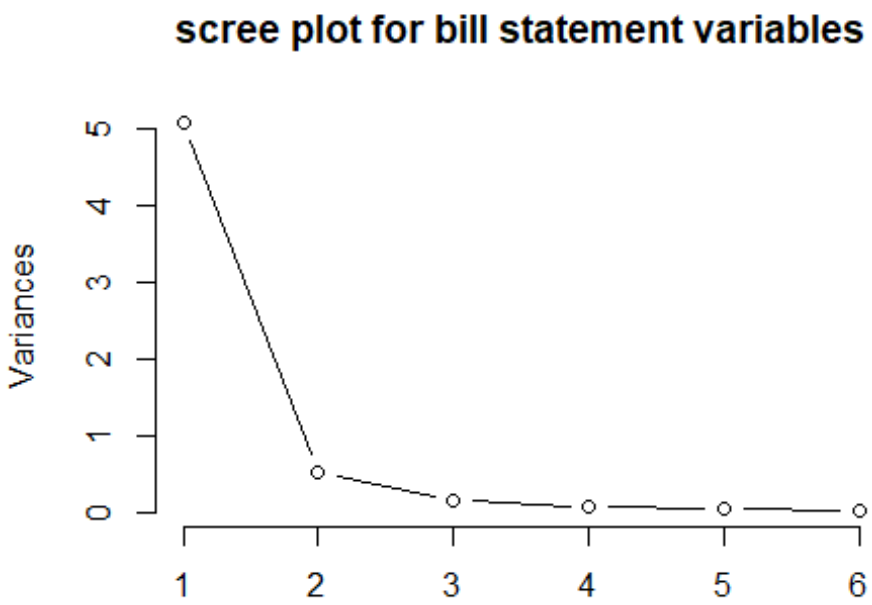
bill statement 와 previous payment 와 관련한 변수들은 각각 상관성이 있을 것으로 보인다. numeric 한 변수들의 상관계수 플랏을 보면,

```
cor(train_imputed[, c(1, 6:17)]) %>% corrplot::corrplot(method = 'circle')
```



bill statement 변수들간의 강한 상관관계가 드러나므로, 이들에 대해 PCA 를 시행해서 차원을 축소할 수 있다. previous payment 와 관련한 변수들의 경우 correlated 된 경향이 잘 보이지는 않지만, PCA 를 통해 적절한 elbow 포인트가 존재한다면 차원을 축소할 수 있을 것이다.

```
# for bill statement variables
pr = prcomp(train_imputed_scale[, c(6:11)])
screeplot(pr, type = 'l', npcs = 6, main = 'scree plot for bill statement variables')
```

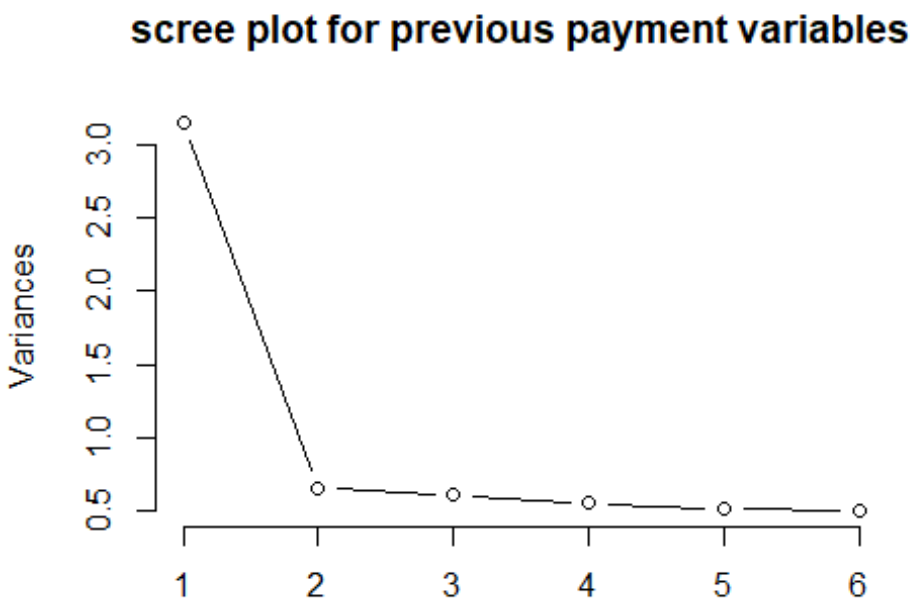


pc1 이 전체 분산의 85%를 설명하므로, pc 1 으로 충분하며, pc score 를 보면 pc1 은 전체의 가중평균의 형태이다. 원래는 이런 적절개수를 CV 를 통해 찾아야하지만, 딱 봤을때 상당히 명확하다고 파악 가능하다.

```
train_imputed_scale$state_pc = pr$x[,'PC1']
train_imputed_scale = train_imputed_scale %>% select(-april_state, -may_state, -june_state, -july_st
ate, -aug_state, -sep_state)
```

previous payment 도 유사한 작업을 거친다.

```
pr2 =  prcomp(train_imputed_scale[, c(6:11)])
screeplot(pr2, type = 'l', npcs = 6, main = 'scree plot for previous payment variables')
```



pc1 이 설명하는 비율은 52%이지만, elobw 와 실제 예측에서 해당변수들을 모두 고려하는 것보다 PC1 만 고려하는 것이 성능측면에서 좋았기 때문에 PC1 을 전체 previous payment 의 대표값처럼 사용한다.

```
train_imputed_scale$pay_pre_pc = pr2$x[,'PC1']
train_imputed_scale = train_imputed_scale %>% select(-april_pay_pre, -may_pay_pre, -june_pay_pre,
                                                     -july_pay_pre, -aug_pay_pre, -sep_pay_pre)
```

EDA & Preprocess 5. 최종 변수들의 확인

```
colnames(train_imputed_scale)

## [1] "credit"    "gender"    "edu"       "martial"   "age"
## [6] "default"   "num_late"  "state_pc"  "pay_pre_pc"
```

기존의 설명변수를 23 개에서 8 개로 줄였으며, 범주형 변수들의 범주도 재범주화를 통해 매우 줄였다. numeric 한 변수들의 상관계수도 0.5 이하였다. 고차항은 state_pc 에 대해 이차항을 GAM 을 통해 고려했었으나, 예측력을 크게 증가시키지 못해 사용하지 않았다.

Modelling 1. CV 구성과 모델 사용

이제 모델링을 진행한다. 사용할 모델은 Randomforest 와 Logistic Regression with Ridge Penalty 이다. 대표적인 data modeling 과 algorithmic modeling 의 방법이고, 예측이 최우선 목적이므로 이 두 모형을 사용했다. LDA 의 경우 범주형 변수들이 있으므로 굳이 사용하지 않았다.

직접 CV 를 나눠서 CV error 를 비교해 최종 모형을 선정하겠다.

```
set.seed(42)
fold_index = createFolds(train_imputed_scale$default, k = 5)
```

randomforest 의 경우 mtry = 4, ntree = 3000 일 때, 오분류율이 0.3340 이었다. Logistic Regression with Ridge Penalty 의 경우 lambda = 0.02212 일 때 0.3055 였다. 파라미터 튜닝 과정은 생략한다.

Modelling 2. 최종 모델

```
ridge_fit = glmnet(x = model.matrix(default ~ ., data = train_imputed_scale)[,-1],
                   y = train_imputed_scale$default, family = 'binomial', alpha = 0, lambda = 0.02212)
```

전체 train set 에 적합함으로써, 최종 모델을 만들었고, 이를 test set 에 대해 평가하는 일만 남았다.

Modelling 3. Test set 에 평가

test set 에 대해 train 과 동일한 전처리를 해주고, 예측을 시행한다.

```
pred_y = predict(ridge_fit, newx = model.matrix(default ~ .,
                                    data = test)[,-1], type = 'response')
pred_y = ifelse(pred_y < 0.5, 0, 1)
sum(abs(test$default - as.numeric(as.character(pred_y)))) / length(pred_y)

## [1] 0.2986667
```

최종 오분류율은 0.298667 로, 정분류율은 0.7014 정도이다.