머신러닝 분반
데이터 분석 대회

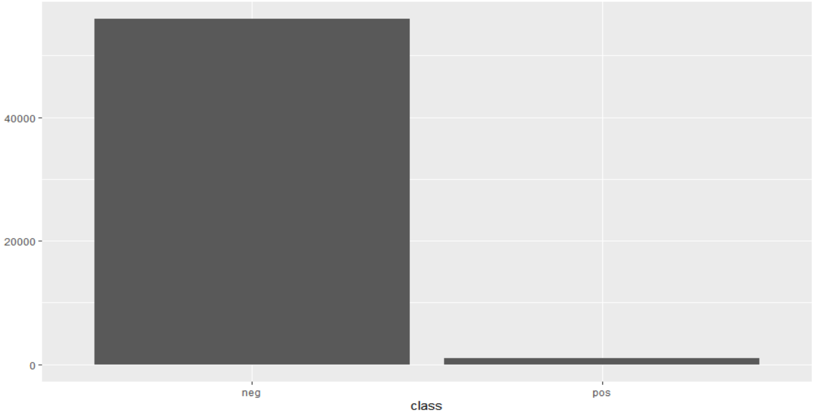
: Scania Truck 부품 별 데이터를 기반으로 한 정비 필요 여부 예측

2조: 김수은, 김창현, 전형록, 조윤경

EDA (Exploratory Data Analysis)

```r
set.seed(1117)

## 전처리
# 데이터 불러오기
# 엑셀에서 na를 R에서는 문자로 인식, 결측치로 인식(na.string)해서 불러오기
train <- read.csv(file = "Train_data.csv", header = TRUE,  na.strings=c("na"))
test <- read.csv(file = "Test_data.csv", header = TRUE,  na.strings=c("na"))


# 데이터 파악
str(train)
str(test)

# train, test 둘 다 1번째열 index임, 제거
train <- subset(train, select = -c(X))
test <- subset(test, select = -c(X))

# factor함수를 사용해서 neg를 0, pos를 1로 바꿈
train$class <- as.factor(train$class)

# train 데이터 확인 #neg(수리X): 55934개, #pos(수리O): 1066개 #불균형 자료임임
qplot((train$class), xlab = "class")
table((train$class))
```

```
> str(train)
'data.frame':	57000 obs. of  171 variables:
 $ class : Factor w/ 2 levels "neg","pos": 1 1 1 1 1 1 1 1 1 1 ...
 $ aa_000: int  41386 29616 241352 8100 2290 34 30768 28662 20896 72800 ...
 $ ab_000: int  NA NA NA NA NA 0 NA NA NA NA ...
 $ ac_000: int  508 1616 NA 86 636 2130706434 2130706432 2130706440 346 282 ...
 $ ad_000: num  488 1490 NA 76 448 44 NA 872 312 234 ...
 $ ae_000: int  0 0 NA 0 0 2 0 0 0 0 ...
 $ af_000: int  0 0 NA 0 0 2 0 0 0 0 ...
 $ ag_000: int  0 0 0 0 0 0 0 NA 0 0 ...
 $ ag_001: int  0 0 0 0 0 0 0 NA 0 0 ...
 $ ag_002: int  0 0 0 0 0 0 0 NA 0 0 ...
 $ ag_003: int  0 0 10140 0 0 0 0 NA 0 0 ...
 $ ag_004: int  51396 452 639334 112 354 5994 157126 NA 194 12274 ...
 $ ag_005: int  886464 42620 9259336 66898 27320 6428 2685830 NA 102298 912008 ...
 $ ag_006: int  1445974 1139952 7148984 400152 77152 1212 1108554 NA 786130 2399834 ...
 $ ag_007: int  463524 594268 676812 66542 31582 0 151266 NA 476456 1166574 ...
 $ ag_008: int  37460 42722 10432 4032 0 0 818 NA 33866 61336 ...
 $ ag_009: int  288 1356 114 0 0 0 0 NA 3394 378 ...
 $ ah_000: int  1201476 782906 6517102 265216 61964 3492 1667578 810572 612814 2086762 ...
 $ ai_000: int  0 0 0 0 0 0 0 0 0 0 ...
 $ aj_000: int  0 0 134 132 0 0 0 444 0 0 ...
 $ ak_000: int  0 0 NA 0 0 0 0 0 0 0 ...
 $ al_000: int  938 0 16102 0 0 1858 74 0 0 0 ...
 $ am_0  : int  2076 0 35412 0 0 2936 432 0 0 0 ...
 $ an_000: int  2413426 1648400 14619728 488114 128728 9798 3108902 1872648 1272020 4036480 ...
 $ ao_000: int  1964298 1444266 12828512 427978 117464 6722 2049320 1627208 1093018 3530046 ...
 $ ap_000: int  569356 279682 3822378 95602 15320 18552 1229742 305260 238328 644504 ...
 $ aq_000: int  403878 166476 1703358 44644 8464 2198 928556 214170 144330 445206 ...
 $ ar_000: int  0 0 NA 0 0 0 0 0 0 0 ...
 $ as_000: int  0 0 0 0 0 0 0 0 0 0 ...
 $ at_000: int  0 0 0 0 0 0 296634 2542 0 0 ...
 $ au_000: int  0 0 0 0 0 0 0 0 0 0 ...
 $ av_000: int  1350 1870 NA 128 102 24 10304 1108 1026 244 ...
```



Handling Missing Value & PCA

```r
# 결측치가 너무 많은 변수를 채워주는 것은 의미없다고 판단, 결측치 30프로 이상 drop
data_sum <- data_sum %>% select_if(colSums(is.na(data_sum))/76000<=0.7)
dim(data_sum) # 170 -> 163 으로 줄어들었음

# 결측치를 변수의 mean으로 대체
data_sum[] <- lapply(data_sum[], function(x) replace(x, is.na(x), mean(x, na.rm = TRUE)))
table(is.na(data_sum)) #결측치 없음
```

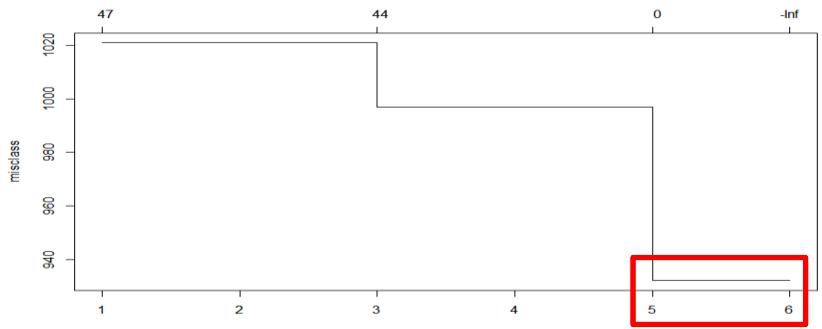
```
> data_sum <- data_sum %>% select_if(colSums(is.na(data_sum))/76000<=0.7)
> dim(data_sum) # 170 -> 163 으로 줄어들었음
[1] 76000   163
>
```

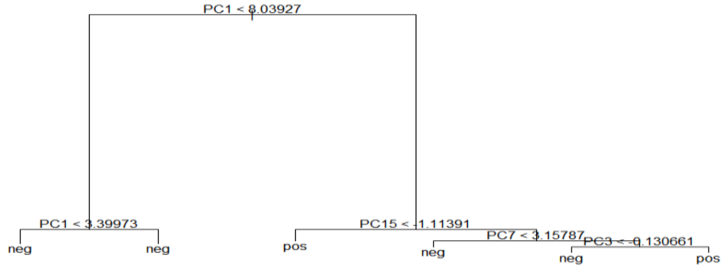
```r
# 163개의 변수는 너무 많다고 판단, PCA를 통해 다중공선성을 줄이기로 함
pca_data_sum <- prcomp(data_sum, center = T, scale. = T) #오류 생김

# constant/zero column이 있다고 나옴, EDA를 통해 보니 cd_000이 constant column으로 나옴
data_sum <- subset(data_sum, select = - c(cd_000)) #cd_000 -> 제거

# 다시 PCA
pca_data_sum <- prcomp(data_sum, center = T, scale. = T)
summary(pca_data_sum) #61번째까지 사용하면 대략 cumulative 89.68%의 설명력을 가짐

# 61개의 변수 사용
pca <- pca_data_sum$x[, 1:61]
summary(pca)
dim(pca) # 19000 + 57000 = 76000
```

```
Importance of components:
                          PC1     PC2     PC3     PC4    PC5     PC6     PC7     PC8
Standard deviation     6.9238 2.79453 2.52213 2.16841 2.0483 1.81284 1.75103 1.73208
Proportion of Variance 0.2959 0.04821 0.03927 0.02902 0.0259 0.02029 0.01893 0.01852
Cumulative Proportion  0.2959 0.34413 0.38339 0.41242 0.4383 0.45860 0.47753 0.49605
                           PC9    PC10    PC11   PC12    PC13    PC14    PC15    PC16
Standard deviation     1.66508 1.55593 1.51794 1.4899 1.41833 1.39743 1.38351 1.33575
Proportion of Variance 0.01711 0.01494 0.01422 0.0137 0.01242 0.01205 0.01182 0.01101
Cumulative Proportion  0.51316 0.52810 0.54233 0.5560 0.56845 0.58050 0.59232 0.60333
                          PC17    PC18    PC19    PC20    PC21   PC22    PC23    PC24
Standard deviation     1.30687 1.29058 1.27838 1.27662 1.25894 1.2274 1.20991 1.16390
Proportion of Variance 0.01054 0.01028 0.01009 0.01006 0.00978 0.0093 0.00904 0.00836
Cumulative Proportion  0.61387 0.62415 0.63424 0.64430 0.65409 0.6634 0.67242 0.68078
                          PC25    PC26    PC27   PC28    PC29    PC30    PC31    PC32
Standard deviation     1.16198 1.14318 1.13402 1.1097 1.09210 1.08307 1.07104 1.06383
Proportion of Variance 0.00833 0.00807 0.00794 0.0076 0.00736 0.00724 0.00708 0.00699
Cumulative Proportion  0.68912 0.69719 0.70512 0.7127 0.72009 0.72733 0.73441 0.74140
                          PC33    PC34    PC35    PC36   PC37    PC38    PC39    PC40
Standard deviation     1.05935 1.05675 1.05428 1.03200 1.0265 1.00158 1.00039 0.99951
Proportion of Variance 0.00693 0.00689 0.00686 0.00657 0.0065 0.00619 0.00618 0.00617
Cumulative Proportion  0.74832 0.75522 0.76208 0.76865 0.7752 0.78135 0.78753 0.79369
                          PC41    PC42    PC43    PC44    PC45    PC46    PC47    PC48
Standard deviation     0.99689 0.98392 0.97633 0.96797 0.95520 0.94824 0.94171 0.92794
Proportion of Variance 0.00613 0.00598 0.00588 0.00578 0.00563 0.00555 0.00547 0.00532
Cumulative Proportion  0.79983 0.80580 0.81169 0.81747 0.82310 0.82865 0.83413 0.83944
                          PC49    PC50    PC51   PC52    PC53    PC54    PC55    PC56
Standard deviation     0.90750 0.89842 0.88940 0.8818 0.87499 0.85226 0.85012 0.83623
Proportion of Variance 0.00508 0.00498 0.00488 0.0048 0.00473 0.00448 0.00446 0.00432
Cumulative Proportion  0.84453 0.84951 0.85439 0.8592 0.86392 0.86840 0.87286 0.87718
                          PC57   PC58    PC59    PC60    PC61    PC62    PC63    PC64
Standard deviation     0.82363 0.8151 0.80439 0.77864 0.77200 0.76172 0.74671 0.73758
Proportion of Variance 0.00419 0.0041 0.00399 0.00374 0.00368 0.00358 0.00344 0.00336
Cumulative Proportion  0.88137 0.8855 0.88946 0.89320 0.89688 0.90046 0.90391 0.90726
```

Decision Tree

```
##############Decision Tree#############
install.packages("caret", dependencies = TRUE)
install.packages("tree")
library(caret)
library(tree)

# cross - validation이나 가지치기 하지 않은 tree 그려보기
# train_y (neg,pos) 예측
treeRaw <- tree(train_y~., data=train_final)
plot(treeRaw)
text(treeRaw)
cv_tree <- cv.tree(treeRaw, FUN = prune.misclass)
plot(cv_tree) #size = 6 일때 최고의 성능임을 알 수 있음.

#decision tree - 가지치기(pruning) 시전
prune_tree <- prune.misclass(treeRaw, best=6)
plot(prune_tree)
text(prune_tree, pretty=0)
```

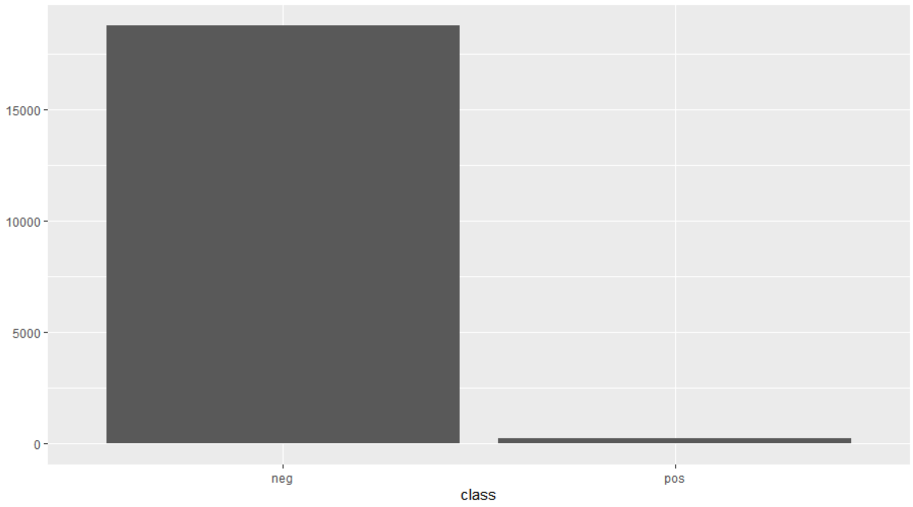




Decision Tree

```
#결과1
#decision tree로 예측
test_x <- as.data.frame(test_x)
pred_class <- predict(prune_tree, test_x, type = "class")
table(pred_class)  #neg: 18752개, pos:248개

pre_class <- as.data.frame(pred_class, col.names="pre_class")

# 처음 원래 데이터에 붙여주기
test <- read.csv(file = "Test_data.csv", header = TRUE,  na.strings=c("na"))
test$pre_class <- pre_class[,1]
tail(test) #확인

write.csv(test,'머신러닝_2조.csv')
```

```
> qplot((test$pre_class), xlab = "class")
> table(pred_class)  #neg: 18752개, pos:248개
pred_class
  neg   pos
18752   248
>
```



Decision Tree

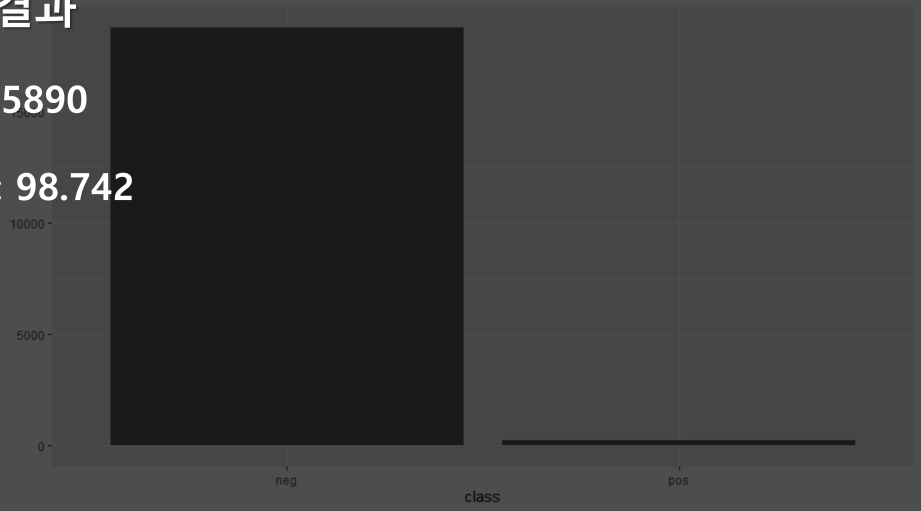
```r
#결과1
#decision tree로 예측
test_x <- as.data.frame(test_x)
pred_class <- predict(prune_tree, test_x, type = "class")
table(pred_class)  #neg: 18752개, pos:248개

pre_class <- as.data.frame(pred_class, col.names="pre_class")

# 처음 원래 데이터에 붙여주기
test <- read.csv(file = "Test_data.csv", header = TRUE,  na.strings=c("na"))
test$pre_class <- pre_class[,1]
tail(test) #확인

write.csv(test,'머신러닝_2조.csv')
```

```
> qplot((test$pre_class), xlab = "class")
> table(pred_class)  #neg: 18752개, pos:248개
pred_class
  neg   pos
18752   248
>
```

예측 결과

Loss: 75890

Accuracy: 98.742

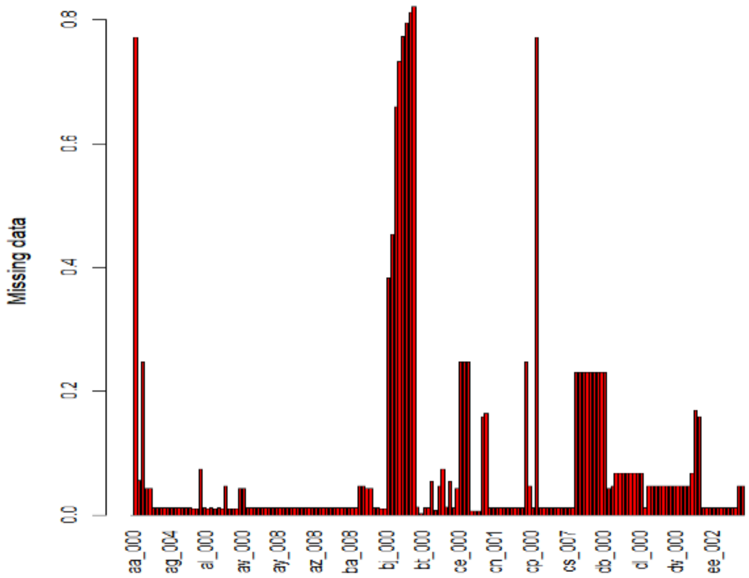


Further improvement: PMM imputation

```r
# 결측치 분포 확인
aggr(data_sum, ylab=c("Missing data","Pattern"))
md.pattern(data_sum)
```

```r
# 결측치 분포가 MAR이기 때문에 MICE 패키지의 pmm을 이용해서 결측치를 채우려 했으나 오류 발생.
# 다중공선성이 높아서 나오는 문제라고 하는데, correlation이 높은 40개 변수를 drop 해봤으나, 실패
data_imp <- mice(data_sum, m=5, maxit=5, meth="pmm", seed=500)
```

```
> data_imp <- mice(data_sum, m=5, maxit=5, meth="pmm", seed=500)

 iter imp variable
  1   1  ab_000Error in solve.default(xtx + diag(pen)) :
  system is computationally singular: reciprocal condition number = 7.67371e-22
>
```



Further improvement: Random Forest Model

```
##############Random Forest#############
ctrl <- trainControl(method = "repeatedcv", repeats=5)
rfFit <- train(train_y~.,
               data=train_final,
               method="rf",
               trControl = ctrl,
               preProcess =c("center","scale"),
               metric = "Accuracy")
rfFit
```


```
##############Random Forest#############
ctrl <- trainControl(method = "repeatedcv", repeats=5)
rfFit <- train(train_y~.,
               data=train_final,
               method="rf",
               trControl = ctrl,
               preProcess =c("center","scale"),
               metric = "Accuracy")
```

감사합니다