투자자를 위한 파산 예측

황성아(2019251126)

Contents

CONTENTS

01 주제 소개

문제상황 01

파산 기업 증가

문제상황 02

다수의 변수

-총 95개의 변수 존재

-투자 전, 기업의 파산 여부를 예측하려고 할 때 고려해야 할 변수의 수가 너무 많음

해결목표

변수 개수 축소를 통한
예측 비용의 감소

한정된 변수만
사용하여
예측 정확도 제고

02 데이터 소개

Company Bankruptcy Prediction

파산 기업 예측

.....

-Bankruptcy data from the Taiwan Economic Journal for the years 1999–2009
-Date created 2021-01-22
-6819개의 row, 96개의 column
-종속변수: 범주형(1: 파산, 0: 파산하지 않음)
독립변수: 수치형

-데이터셋 획득 경로
https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction

02 데이터 소개

95개의 변수

변수 카테고리	해당 변수	타입	형태
ROA	x1 ~ x3	Num	수치형
이익률	X4 ~ x10	Num	수치형
영업활동	x11, x42 ~ x48, x50, x52 ~ x57, x69 ~ x74, x82, x86 ~ x90, x93	Num	수치형
투자활동	x12, x49, x76	Num	수치형
주당	x16 ~ x23	Num	수치형
성장률	x24 ~ 31	Num	수치형
재무활동	x32 ~ x41, x68, x78, x91 ~ x92, x95	Num	수치형
부채	x58 ~ x67, x77, x79, x84	Num	수치형
현금 흐름	x13, x75, x80 ~ x81, x83	Num	수치형
파산 여부	Y	Int	범주형
부채/자산 비교 지표	x85	Int	범주형
순이익 지표	x94	Int	범주형

02 데이터 소개

95개의 변수

변수 카테고리	해당 변수	타입	형태
ROA	x1 ~ x3	Num	수치형
이익률	X4 ~ x10	Num	수치형
영업활동	x11, x42 ~ x48, x50, x52 ~ x57, x69 ~ x74, x82, x86 ~ x90, x93	Num	수치형
투자활동	x12, x49, x76	Num	수치형
주당	x16 ~ x23	Num	수치형
성장률	x24 ~ 31	Num	수치형
재무활동	x32 ~ x41, x68, x78, x91 ~ x92, x95	Num	수치형
부채	x58 ~ x67, x77, x79, x84	Num	수치형
현금 흐름	x13, x75, x80 ~ x81, x83	Num	수치형
파산 여부	Y 기업이 파산하면 1, 아니면 0	Int	범주형
부채/자산 비교 지표	x85	Int	범주형
순이익 지표	x94	Int	범주형

Outcome variable

03 데이터분석 및 전처리

(1) 전처리

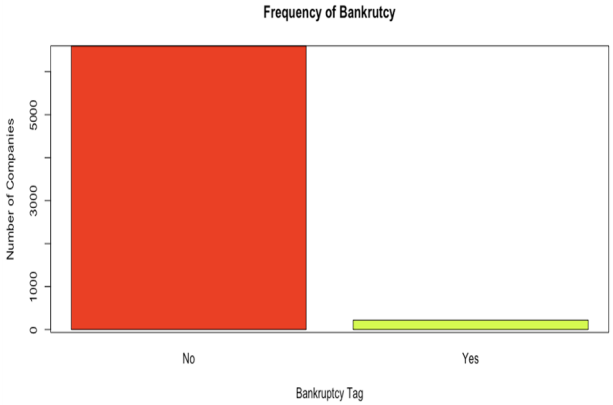
```
##### 데이터 #####

# 읽어오기
bk <- read.csv("bankrupcy.csv")
head(bk)
str(bk)
summary(bk)

# net income flag, Liability Assets flag 삭제
table(bk$Net.Income.Flag)
table(bk$Liability.Assets.Flag)

bk <- bk[,-c(86,95)]
head(bk)
dim(bk)

# 범주형 변수 변환 (분류대상)
bk$Bankrupt. <- factor(bk$Bankrupt.)
table(bk$Bankrupt.)

# 변수명 변경 x1.aaa
colnames(bk)[1] <- paste("Y.", colnames(bk)[1], sep="")
class(colnames(bk)[1])
head(colnames(bk[1]))

for (i in 2:dim(bk)[2]){
  colnames(bk)[i] <- paste("X", i-1, ".", colnames(bk[i]), sep="")
}

colnames(bk)
```

```
> table(bk$Net.Income.Flag)

   1
6819
> table(bk$Liability.Assets.Flag)

   0    1
6811    8
```

- Net Income Flag는 값이 모두 1
  → 분석에서 제외
- Liability Assets Flag 분포 불균형이 심함
  → 분석에서 제외

03 데이터분석 및 전처리

(2) Outcome 변수 분포 불균형 해소



OUTCOME 변수 분포의 불균형 해결

- Random over sampling, SMOTE 등
- 데이터의 분포가 작은 클래스의 값을 분포가 큰 클래스로 맞추는 방법

정보의 손실을 막기 위해 Oversampling 방법 이용

Random Over Sampling

분포가 적은 record를 랜덤으로 복제하여 분포 불균형을 해결하는 방법

SMOTE

분포가 적은 데이터의 샘플을 취한 뒤 최근접 이웃과 샘플의 차에 0 ~ 1 사이의 임의의 값을 곱하여 훈련 데이터에 추가하는 방법

03 데이터분석 및 전처리

(2) 분포 불균형 해소

```
##### oversampling #####
table(bk$Y.Bankrupt.) # 약 30배
```

```
# caret upsampling - 개수 똑같이 + record 중복
library(caret)
bk.os.up <- upSample(subset(bk, select=-Y.Bankrupt.), bk$Y.Bankrupt.)

table(bk.os.up$Class)
colnames(bk.os.up)[94] <- "Y.Bankrupt."
str(bk.os.up)
bk.os.up <- bk.os.up[c(94,1:93)]
head(bk.os.up)
```

```
> table(bk.os.up$Class)

   0    1
6599 6599
```

```
# SMOTE - KNN=5 기반
?smote
install.packages("performanceEstimation")
library(performanceEstimation)

bk.os.sm <- smote(Y.Bankrupt. ~ ., bk, perc.over=30, perc.under=1)
table(bk.os.sm$Y.Bankrupt.)
head(bk.os.sm)
```

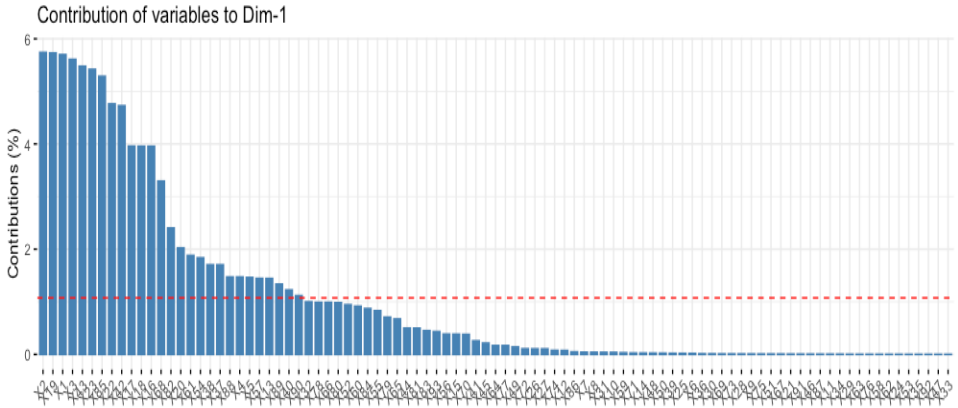
```
> table(bk.os.sm$Y.Bankrupt.)

   0    1
6600 6820
```

```
# csv 파일로 저장
write.csv(bk.os.up, file="bankrupt_ROS.csv", row.names=FALSE)
write.csv(bk.os.sm, file="bankrupt_SMOTE.csv", row.names=FALSE)
```

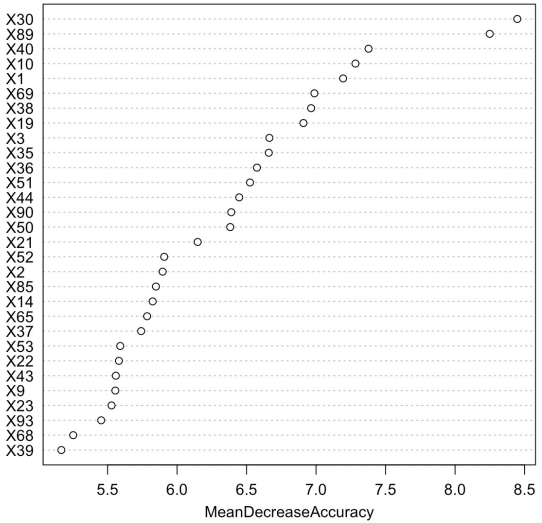
03 데이터분석 및 전처리

(3) 변수 축소

PCA와 Variance Importance Plot을 통한 1차 변수 축소를 진행



PCA



VIP

03 데이터분석 및 전처리

(3) 변수 축소

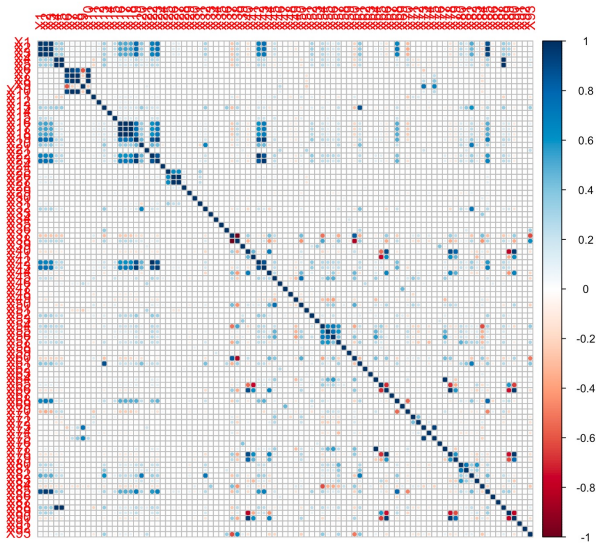
PCA와 VIP 상의 순위를 백분위화 시키고,
점수화 과정(PCA백분위 + VIP백분위 * 1.5)을 거쳐
변수 우선순위를 도출함

PCA	VIP		SCORE	
X2	X30	Score = PCA백분위 * 1 + VIP백분위 * 1.5	X1	214
X19	X89		X19	203
X1	X40		X3	190
X3	X10		X2	156
X43	X1		X38	148
X23	X69		X89	147
X85	X38		X30	145
X22	X19		X40	139
X42	X3		X10	130
X17	X35		X85	129
X18	X36		X69	120
X16	X51		X43	106
X68	X44		X22	100
X82	X90		X35	100
X20	X50		X36	95
X61	X21		X23	93
X54	X52		X51	90
X38	X2		X44	85
X37	X85		X90	81
X88	X14		X50	75
X4	X65		X21	70
X5	X37		X37	70
X57	X53		X42	67
X13	X22		X52	65
X89	X43		X17	63
X40	X9		X18	59
X90	X23		X68	57
	X93		X16	56
	X68		X14	50
	X39		X82	48
			X65	45
			X20	44
			X61	41
			X54	37
			X53	35
			X88	26
			X4	22
			X9	20

03 데이터분석 및 전처리

(3) 변수 축소

그 후 다중공산성 해소 및 추가적인 변수 축소를 위해
상관관계 분석 후 상관관계 군(Correlation group)을 형성,
대표값을 취하는 방식으로 추가적으로 변수를 축소함



X3 ~ X2 ~ X1 ~ X19 ~ X23 ~ X43

X5 ~ X4 ~ X88

X8 ~ X7 ~ X6

X18 ~ X17 ~ X16 ~ X19 ~ X23 ~ X42 ~X43

X27 ~ X26

X38 ~ X37 ~ X60

X73 ~ X9 ~ X75

X50 ~ X45

X56 ~ X48

X61 ~ X13

X65 ~ X41

X66 ~ X40 ~ X78 ~ X89 ~ X90

X68 ~ X2

X77 ~ X64

X79 ~ X40

03 데이터분석 및 전처리

(3) 변수 축소

총 12개의 변수가 선택되었으며,

Default Tree를 통해 accuracy를 측정해보니

변수 축소 이후의 accuracy가 오히려 상승한 것으로 보아 잘 축소되었다고 판단함

축소 이전

```
          Accuracy : 0.9648
            95% CI : (0.9572, 0.9714)
No Information Rate : 0.9707
P-Value [Acc > NIR] : 0.966530

             Kappa : 0.2445
```

축소 이후

```
          Accuracy : 0.9754
            95% CI : (0.9689, 0.9809)
No Information Rate : 0.9707
P-Value [Acc > NIR] : 0.07514

             Kappa : 0.3284
```

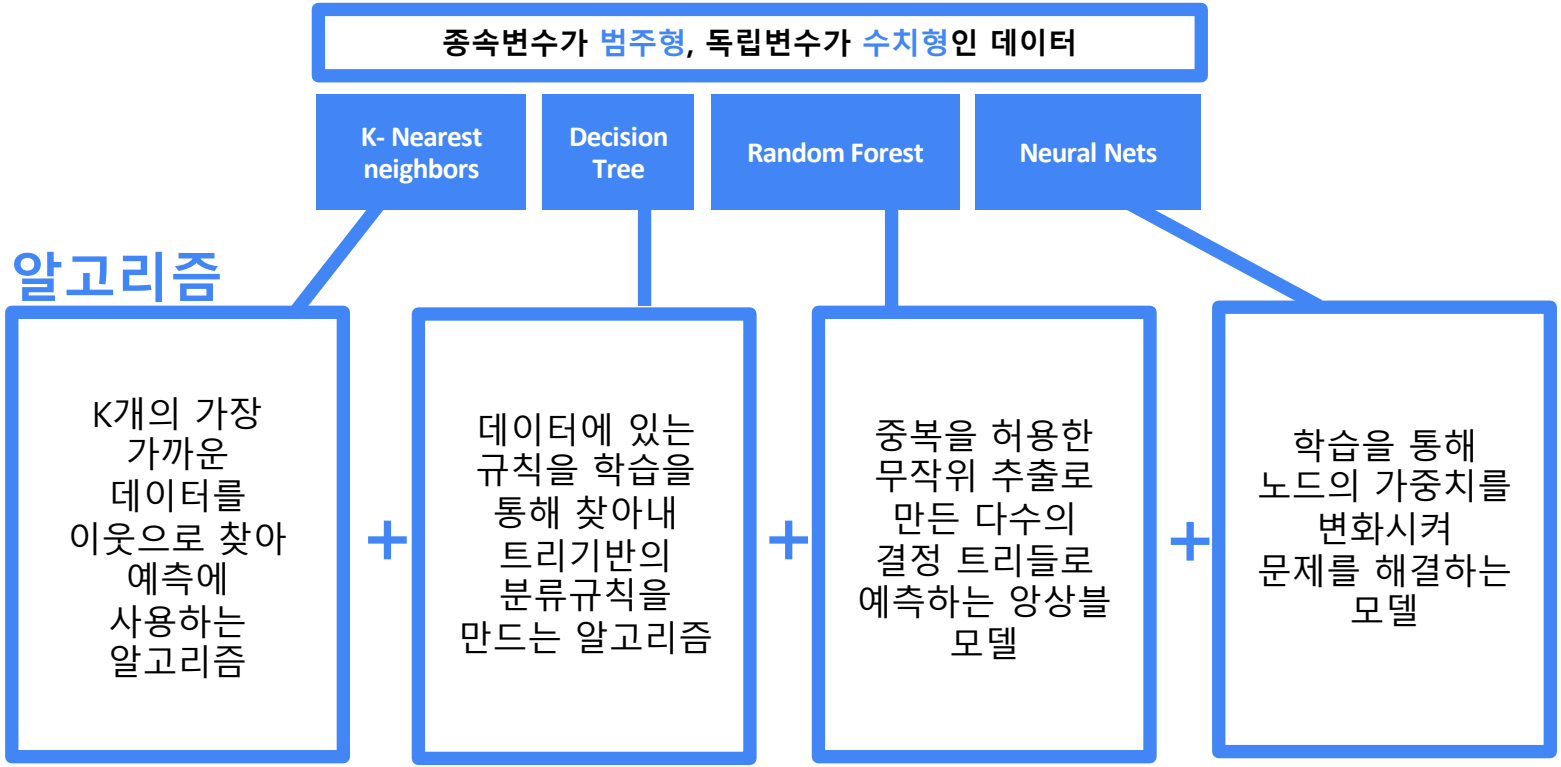
03 데이터분석 및 전처리

(3) 변수 축소

선택된 변수 목록

X1 : ROA.C..before.interest.and.depreciation.before.interest
X10 : Continuous.interest.rate..after.tax
X22 : Operating.Profit.Per.Share.
X23 : Per.Share.Net.profit.before.tax
X30 : Net.Value.Growth.Rate
X35 : Interest.Expense.Ratio
X36 : Total.debt.Total.net.worth
X38 : Net.worth.Assets
X51 : Revenue.per.person
X69 : Total.income.Total.expense
X85 : Net.Income.to.Total.Assets
X89 : Net.Income.to.Stockholder.s.Equity

04 데이터 마이닝 기법 적용



04 데이터 마이닝 기법 적용

(1) train/valid/test data set

ROS

```
## ROS final data
bankrupt_ROS <- read.csv("bankrupt_ROS.csv", na.strings = "", header = T)
bankrupt_ROS_final <- bankrupt_ROS[,c(1,2,11,23,24,31,36,37,39,52,70,86,90)]
dim(bankrupt_ROS_final)
head(bankrupt_ROS_final)
bankrupt_ROS_final$Y.Bankrupt. <- factor(bankrupt_ROS_final$Y.Bankrupt.)
```

선택된 column 추출

```
## modeling - ROS
set.seed(99)
spl <- sample(c(1:3), size=nrow(bankrupt_ROS_final), replace=TRUE, prob=c(0.6,0.2,0.2))
train.ros <- bankrupt_ROS_final[spl==1,]
train.index.ros <- rownames(bankrupt_ROS_final)[spl==1]
valid.ros <- bankrupt_ROS_final[spl==2,]
valid.index.ros <- rownames(bankrupt_ROS_final)[spl==2]
test.ros <- bankrupt_ROS_final[spl==3,]
test.index.ros <- rownames(bankrupt_ROS_final)[spl==3]

dim(train.ros)  # 7974, 13
dim(valid.ros) # 2639, 13
dim(test.ros) # 2585, 13
```

Data partition

```
# Normalization
train.ros.norm <- train.ros
valid.ros.norm <- valid.ros
test.ros.norm <- test.ros

library(caret)
head(train.ros)
norm.values <- preProcess(train.ros[,c(2:13)], method=c("center","scale"))

train.ros.norm[,c(2:13)] <- predict(norm.values, train.ros[,c(2:13)])
valid.ros.norm[,c(2:13)] <- predict(norm.values, valid.ros[,c(2:13)])
test.ros.norm[,c(2:13)] <- predict(norm.values, test.ros[,c(2:13)])
```

정규화

SMOTE

```
## SMOTE final data
bankrupt_SMOTE <- read.csv("bankrupt_SMOTE.csv", na.strings = "", header = T)
bankrupt_SMOTE_final <- bankrupt_SMOTE[,c(1,2,11,23,24,31,36,37,39,52,70,86,90)]
dim(bankrupt_SMOTE_final)
head(bankrupt_SMOTE_final)
bankrupt_SMOTE_final$Y.Bankrupt. <- factor(bankrupt_SMOTE_final$Y.Bankrupt.)
```

```
## modeling - SMOTE
set.seed(99)
spl <- sample(c(1:3), size=nrow(bankrupt_SMOTE_final), replace=TRUE, prob=c(0.6,0.2,0.2))

train.sm <- bankrupt_SMOTE_final[spl==1,]
valid.sm <- bankrupt_SMOTE_final[spl==2,]
test.sm <- bankrupt_SMOTE_final[spl==3,]

dim(train.sm)  # 8113, 13
dim(valid.sm) # 2674, 13
dim(test.sm) # 2633, 13
```

```
# Normalization
train.sm.norm <- train.sm
valid.sm.norm <- valid.sm
test.sm.norm <- test.sm

library(caret)
head(train.sm)
norm.values <- preProcess(train.sm[,c(2:13)], method=c("center","scale"))

train.sm.norm[,c(2:13)] <- predict(norm.values, train.sm[,c(2:13)])
train.index.sm <- rownames(bankrupt_SMOTE_final)[spl==1]
valid.sm.norm[,c(2:13)] <- predict(norm.values, valid.sm[,c(2:13)])
valid.index.sm <- rownames(bankrupt_SMOTE_final)[spl==2]
test.sm.norm[,c(2:13)] <- predict(norm.values, test.sm[,c(2:13)])
test.index.sm <- rownames(bankrupt_SMOTE_final)[spl==3]
```

04 데이터 마이닝 기법 적용

(1) train/valid/test data set (Neural Net)

ROS

```
# Neural Nets
scale.values <- caret::preProcess(bankrupt_ROS_final[,c(2:13)],
                                  rangeBounds = c(0,1), methods = "range")
scaled = predict(scale.values, bankrupt_ROS_final[,c(2:13)])

bankrupt_ROS_scaled <- cbind(Y.Bankrupt. = bankrupt_ROS_final$Y.Bankrupt., scaled)

bankrupt_ROS_scaled[1] <- as.numeric(bankrupt_ROS_scaled$Y.Bankrupt.)-1
head(bankrupt_ROS_scaled[1])

var <- colnames(bankrupt_ROS_scaled)[-1]
var
```

범주 수정

```
library(nnet)
head(class.ind(bankrupt_ROS_scaled[train.index.ros,]$Y.Bankrupt.))

train.nn.ros <- cbind(bankrupt_ROS_scaled[train.index.ros,var],
                      class.ind(bankrupt_ROS_scaled[train.index.ros,]$Y.Bankrupt.))
names(train.nn.ros) <- c(var, paste("Y.Bankrupt._", c(0,1), sep=""))
head(train.nn.ros)

valid.nn.ros <- cbind(bankrupt_ROS_scaled[valid.index.ros,var],
                      class.ind(bankrupt_ROS_scaled[valid.index.ros,]$Y.Bankrupt.))
names(valid.nn.ros) <- c(var, paste("Y.Bankrupt._", c(0,1), sep=""))
head(valid.nn.ros)

test.nn.ros <- cbind(bankrupt_ROS_scaled[test.index.ros,var],
                     class.ind(bankrupt_ROS_scaled[test.index.ros,]$Y.Bankrupt.))
names(test.nn.ros) <- c(var, paste("Y.Bankrupt._", c(0,1), sep=""))
head(test.nn.ros)
```

범주형 변수 Class.ind

SMOTE

```
# Neural Nets
scale.values <- caret::preProcess(bankrupt_SMOTE_final[,c(2:13)],
                                  rangeBounds = c(0,1), methods = "range")
scaled = predict(scale.values, bankrupt_SMOTE_final[,c(2:13)])

bankrupt_SMOTE_scaled <- cbind(Y.Bankrupt. = bankrupt_SMOTE_final$Y.Bankrupt., scaled)

bankrupt_SMOTE_scaled[1] <- as.numeric(bankrupt_SMOTE_scaled$Y.Bankrupt.)-1
head(bankrupt_SMOTE_scaled[1])

var <- colnames(bankrupt_SMOTE_scaled)[-1]
var
```

```
library(nnet)
head(class.ind(bankrupt_SMOTE_scaled[train.index.sm,]$Y.Bankrupt.))

train.nn.sm <- cbind(bankrupt_SMOTE_scaled[train.index.sm,var],
                     class.ind(bankrupt_SMOTE_scaled[train.index.sm,]$Y.Bankrupt.))
names(train.nn.sm) <- c(var, paste("Y.Bankrupt._", c(0,1), sep=""))
head(train.nn.sm)

valid.nn.sm <- cbind(bankrupt_SMOTE_scaled[valid.index.sm,var],
                     class.ind(bankrupt_SMOTE_scaled[valid.index.sm,]$Y.Bankrupt.))
names(valid.nn.sm) <- c(var, paste("Y.Bankrupt._", c(0,1), sep=""))
head(valid.nn.sm)

test.nn.sm <- cbind(bankrupt_SMOTE_scaled[test.index.sm,var],
                    class.ind(bankrupt_SMOTE_scaled[test.index.sm,]$Y.Bankrupt.))
names(test.nn.sm) <- c(var, paste("Y.Bankrupt._", c(0,1), sep=""))
head(test.nn.sm)
```

04 데이터 마이닝 기법 적용

(2) KNN

```r
# KNN
library(FNN)

accuracy.df <- data.frame(k = seq(1, 12, 1), accuracy = rep(0, 12))
for (i in 1:12){
  knn.pred <- FNN::knn(train = train.ros.norm[, 2:13], test = valid.ros.norm[, 2:13],
                       cl = train.ros.norm[,1], k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid.ros.norm[,1])$overall[1]
}
accuracy.df

knn.pred2 <- FNN::knn(train = train.ros.norm[,2:13], test = test.ros.norm[,2:13],
                      cl = train.ros.norm[,1], k = 2)

res.df <- data.frame(test.ros, knn.pred2)
head(res.df, n = 5)
confusionMatrix(knn.pred2, test.ros.norm[,1])
```

```
               Accuracy : 0.9857
                 95% CI : (0.9803, 0.9899)
    No Information Rate : 0.5041
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9714

 Mcnemar's Test P-Value : 3.252e-09

            Sensitivity : 0.9711
```

ROS

```r
# KNN
library(FNN)

accuracy.df <- data.frame(k = seq(1, 12, 1), accuracy = rep(0, 12))
for (i in 1:12){
  knn.pred <- FNN::knn(train = train.sm.norm[, 2:13], test = valid.sm.norm[, 2:13],
                       cl = train.sm.norm[,1], k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid.sm.norm[,1])$overall[1]
}
accuracy.df

knn.pred1 <- FNN::knn(train = train.sm.norm[,2:13], test = test.sm.norm[,2:13],
                      cl = train.sm.norm[,1], k = 1)

res.df <- data.frame(test.sm, knn.pred1)
head(res.df, n = 5)

confusionMatrix(knn.pred1, test.sm.norm[,1])
```

```
               Accuracy : 0.9609
                 95% CI : (0.9528, 0.968)
    No Information Rate : 0.5131
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9216

 Mcnemar's Test P-Value : 2.996e-07

            Sensitivity : 0.9392
```

SMOTE

04 데이터 마이닝 기법 적용

(3) Default Tree

```
# Default tree
library(rpart)
library(rpart.plot)
default.ct <- rpart(Y.Bankrupt. ~ .,data = train.ros.norm, method = "class")
print(default.ct)
prp(default.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
    box.col = ifelse(default.ct$frame$var == "<leaf>", 'gray', 'white'))

default.pred <- predict(default.ct, test.ros.norm, type = "class")
head(default.pred, n = 30)

confusionMatrix(default.pred, test.ros.norm$Y.Bankrupt.)
```

```
          Accuracy : 0.8855
            95% CI : (0.8726, 0.8975)
No Information Rate : 0.5041
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.7708

Mcnemar's Test P-Value : 4.995e-10

          Sensitivity : 0.8424
```

ROS

```
# Default tree
library(rpart)
library(rpart.plot)
default.ct <- rpart(Y.Bankrupt. ~ .,data = train.sm.norm, method = "class")
print(default.ct)
prp(default.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,
    box.col = ifelse(default.ct$frame$var == "<leaf>", 'gray', 'white'))

default.pred <- predict(default.ct, test.sm.norm, type = "class")
head(default.pred, n = 30)

confusionMatrix(default.pred, test.sm.norm$Y.Bankrupt)
```

```
          Accuracy : 0.8982
            95% CI : (0.886, 0.9095)
No Information Rate : 0.5131
P-Value [Acc > NIR] : <2e-16

             Kappa : 0.7963

Mcnemar's Test P-Value : 1

          Sensitivity : 0.8955
```

SMOTE

04 데이터 마이닝 기법 적용

(4) Full Tree

```
# Full Tree
library(rpart)
library(rpart.plot)
deeper.ct <- rpart(Y.Bankrupt. ~ ., data = train.ros.norm, method = "class",
                   cp = 0, minsplit = 1)
prp(deeper.ct, type = 1, extra = 1, under = TRUE, split.font = 1,
    varlen = -10, box.col = ifelse(deeper.ct$frame$var == "<leaf>",
                                   'gray', 'white'))
deeper.pred <- predict(deeper.ct, test.ros.norm, type = "class")

confusionMatrix(deeper.pred, test.ros.norm$Y.Bankrupt.)
```

```
                Accuracy : 0.9841
                  95% CI : (0.9785, 0.9886)
    No Information Rate : 0.5041
    P-Value [Acc > NIR] : < 2.2e-16

                   Kappa : 0.9683

 Mcnemar's Test P-Value : 4.185e-10

            Sensitivity : 0.9680
```

ROS

```
# Full Tree
library(rpart)
library(rpart.plot)
deeper.ct <- rpart(Y.Bankrupt. ~ ., data = train.sm.norm, method = "class",
                   cp = 0, minsplit = 1)
prp(deeper.ct, type = 1, extra = 1, under = TRUE, split.font = 1,
    varlen = -10, box.col = ifelse(deeper.ct$frame$var == "<leaf>",
                                   'gray', 'white'))
deeper.pred <- predict(deeper.ct, test.sm.norm, type = "class")

confusionMatrix(deeper.pred, test.sm.norm$Y.Bankrupt.)
```

```
                Accuracy : 0.9419
                  95% CI : (0.9323, 0.9505)
    No Information Rate : 0.5131
    P-Value [Acc > NIR] : <2e-16

                   Kappa : 0.8837

 Mcnemar's Test P-Value : 0.4188

            Sensitivity : 0.9360
```

SMOTE

04 데이터 마이닝 기법 적용

(5) Random Forest

```r
# Random Forest
library(randomForest)
rf <-  randomForest(Y.Bankrupt. ~., data = train.ros.norm,
                    importance = TRUE)
rf.pred <- predict(rf, test.ros.norm)
head(rf.pred, n = 30)

confusionMatrix(rf.pred, test.ros.norm$Y.Bankrupt.)
```

```
          Accuracy : 0.9946
            95% CI : (0.9909, 0.997)
No Information Rate : 0.5041
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.9892

Mcnemar's Test P-Value : 0.000512

        Sensitivity : 0.9891
```

ROS

```r
# Random Forest
library(randomForest)
rf <-  randomForest(Y.Bankrupt. ~., data = train.sm.norm,
                    importance = TRUE)
rf.pred <- predict(rf, test.sm.norm)
head(rf.pred, n = 30)

confusionMatrix(rf.pred, test.sm.norm$Y.Bankrupt.)
```

```
          Accuracy : 0.9757
            95% CI : (0.9691, 0.9812)
No Information Rate : 0.5131
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.9513

Mcnemar's Test P-Value : 0.001778

        Sensitivity : 0.9649
```

SMOTE

04 데이터 마이닝 기법 적용

(6) Boosting

```
# boosting
library(adabag)

boost <- boosting(Y.Bankrupt. ~ ., data = train.ros.norm)
bt.pred <- predict(boost, test.ros.norm)
head(bt.pred$class, n =20)
class(bt.pred$class)

confusionMatrix(factor(bt.pred$class), test.ros.norm$Y.Bankrupt.)
```

```
          Accuracy : 0.9903
            95% CI : (0.9858, 0.9937)
No Information Rate : 0.5041
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.9807

Mcnemar's Test P-Value : 1.587e-06

        Sensitivity : 0.9805
```

ROS

```
# boosting
library(adabag)

boost <- boosting(Y.Bankrupt. ~ ., data = train.sm.norm)
bt.pred <- predict(boost, test.sm.norm)
head(bt.pred$class, n =20)
class(bt.pred$class)

confusionMatrix(factor(bt.pred$class), test.sm.norm$Y.Bankrupt.)
```

```
          Accuracy : 0.9681
            95% CI : (0.9607, 0.9745)
No Information Rate : 0.5131
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.9361

Mcnemar's Test P-Value : 0.006377

        Sensitivity : 0.9571
```

SMOTE

04 데이터 마이닝 기법 적용

(7) Neural Net

```
library(neuralnet)
nn1 <- neuralnet(Y.Bankrupt._0 + Y.Bankrupt._1 ~.,data = train.nn.ros, hidden = 6, stepmax = 1e+06)
plot(nn1)

head(train.nn.ros)
test.prediction <- compute(nn1, test.nn.ros[,-c(13:14)])
head(test.prediction)
test.class <- apply(test.prediction$net.result, 1, which.max) -1
head(test.class, n = 50)
class(test.class)

head(test.ros)
confusionMatrix(factor(test.class), factor(ifelse(test.nn.ros$Y.Bankrupt._0 == 1, 0, 1)))
```

```
               Accuracy : 0.8863
                 95% CI : (0.8734, 0.8983)
    No Information Rate : 0.5041
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7725

 Mcnemar's Test P-Value : 0.1798

            Sensitivity : 0.8760
```

ROS

```
library(neuralnet)
nn1 <- neuralnet(Y.Bankrupt._0 + Y.Bankrupt._1 ~.,data = train.nn.sm, hidden = 6)
plot(nn1)

head(train.nn.ros)
test.prediction <- compute(nn1, test.nn.sm[,-c(13:14)])
head(test.prediction)
test.class <- apply(test.prediction$net.result, 1, which.max) -1
head(test.class, n = 50)
class(test.class)

confusionMatrix(factor(test.class), factor(ifelse(test.nn.sm$Y.Bankrupt._0 == 1, 0, 1)))
```

```
               Accuracy : 0.8921
                 95% CI : (0.8797, 0.9037)
    No Information Rate : 0.5131
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7838

 Mcnemar's Test P-Value : 4.898e-06

            Sensitivity : 0.8588
```

SMOTE

05 결론 및 실적용

(1) 최적 모델 선정

정확도/재현율	KNN	DT	FT	RF	Boosting	NN
ROS	0.9857/0.9711 (k=2)	0.8855/0.8424	0.9841/0.9680	0.9946/0.9891	0.9903/0.9805	0.8863/0.8760(6)
SM	0.9609/0.9392 (k=1)	0.8982/0.8955	0.9419/0.9360	0.9757/0.9680	0.9681/0.9571	0.8921/0.8729(6)

Random Over Sampling 과 SMOTE 두 가지의
불균형 해결 방법을 이용해본 결과,
대체적으로 Random Over Sampling에서 더 좋은 성능을 보임

별도의 파라미터를 조정하지 않은 상태에서
Random Over Sampling 방법을 적용한
Random Forest 알고리즘이 가장 좋은 성능을 보임

↓

Random Forest(Ros)의 파라미터 조정을 통해 최적의 모델을 도출하고자 함

05 결론 및 실적용

(1) 최적 모델 선정

	Accuracy	Sensitivity
Case1 ntree =2000, mtry =2	0.9946	0.9891
Case2 ntree =2000, mtry =5	0.993	0.9860
best Case3 ntree =3000, mtry =2	0.9954	0.9906
Case4 ntree =3000, mtry =5	0.9934	0.9867

ntree: 트리 개수, mtry: 변수 개수

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1270    0
         1   12 1303

               Accuracy : 0.9954
                 95% CI : (0.9919, 0.9976)
    No Information Rate : 0.5041
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9907

 Mcnemar's Test P-Value : 0.001496

            Sensitivity : 0.9906
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9909
             Prevalence : 0.4959
         Detection Rate : 0.4913
   Detection Prevalence : 0.4913
      Balanced Accuracy : 0.9953

       'Positive' Class : 0
```

Random Forest(ros)의 중요 파라미터를 조정시켜본 결과,
ntree =3000, mtry =2일 때 정확도와 재현율이 모두 가장 많이 증가하여 최적 모델로 선정

05 결론 및 실적용

(2) 실제 기업에 적용

먼저 시가총액 순서로 20개의 국내 기업들의 변수를 정리
전자공시시스템 DART에서 각 기업별 보고서를 참고하여 정리
2021년 3분기말 기준의 재무제표와 손익계산서 내용을 참고

05 결론 및 실적용

(2) 실제 기업에의 적용

최종적으로 20개 기업의 변수 도출

변수	삼성 전자	SK 하이 닉스	네이버	삼성 바이오 로직스	카카오	LG 화학	삼성 SDI	현대 자동차	기아	카카오 뱅크	셀트 리온	카카오 페이	크래 프톤	포스코	KB 금융	현대 모비스	SK 바이오 사이 언스	삼성 물산	LG 전자	SK 이노 베이션
X1	0.029953	0.039814	0.009791	0.017352	0.047516	0.013646	0.017145	0.006434	0.017594	0.004729	0.025369	-0.001122	0.033617	0.029152	0.005839	0.01074	0.039828	0.006964	0.009466	0.010436
X10	0.166173	0.280827	0.186843	0.292353	0.497639	0.064079	0.122224	0.051507	0.063915	0.222205	0.350242	-0.016413	0.341632	0.127326	0.312229	0.053609	0.346138	0.045177	0.027492	0.041539
X22	0.002329	0.005731	0.002129	0.00253	0.000378	0.009282	0.005307	0.005802	0.003274	0.000431	0.001189	-7.80E-06	0.003989	0.035748	0.012525	0.004827	0.001312	0.000748	0.00299	0.0066
X23	0.002408	0.006317	0.003014	0.002981	0.002407	0.009421	0.007824	0.006994	0.003962	0.000431	0.00136	-1.45E-05	0.005515	0.038531	0.012593	0.007643	0.001262	0.002865	0.003489	0.007508
X30	0.075442	0.131101	1.932216	0.068331	0.46817	0.206922	0.113454	0.078208	0.130124	0.958651	0.141013	0.449658	2.722938	0.122638	0.098411	0.044866	4.384006	-0.01111	0.157507	0.249911
X35	0.022901	0.056507	0.135641	0.08756	0.018577	0.252218	0.087129	0.954292	0.107476	0.683252	0.085493	-0.128314	0.050103	0.130824	0.758595	0.045522	0.034957	0.212783	0.46422	0.619387
X36	0.382977	0.418204	0.361692	0.545749	0.671798	1.196872	0.648536	1.80775	0.909134	5.481887	0.404118	4.677784	0.17349	0.684097	12.645	0.431904	0.351301	0.653293	1.686997	1.533947
X38	0.723078	0.705117	0.734381	0.646935	0.598159	0.455193	0.606599	0.356157	0.523798	0.154276	0.712191	0.176125	0.852159	0.59379	0.073287	0.698371	0.740028	0.604853	0.372163	0.394641
X51	649.1395	398.3978	582.9413	120.3559	583.9581	795.2446	303.0387	402.8134	504.0821	752.0318	185.7142	186.2399	334.1479	1143.571	80562.67	940.1365	197.1679	945.1382	481.7708	4389.914
X69	1.199289	1.390486	1.229775	1.413133	1.990599	1.068466	1.139242	1.054305	1.068279	1.285686	1.539035	0.983852	1.518907	1.145904	1.453972	1.056645	1.529375	1.047314	1.028269	1.043339
X85	0.029953	0.039814	0.009791	0.017352	0.047516	0.013646	0.017145	0.006434	0.017594	0.004729	0.025369	-0.001122	0.033617	0.029152	0.005839	0.01074	0.039828	0.006964	0.009466	0.010436
X89	0.041424	0.056464	0.013333	0.026821	0.079437	0.029979	0.028265	0.018064	0.033588	0.030656	0.035621	-0.006373	0.039449	0.049095	0.079672	0.015379	0.05382	0.011514	0.025436	0.026444

05 결론 및 실적용

(2) 실제 기업에 적용

모델에 적용하여 20개 기업의 파산 위험 파악

```
#test data
library(readxl)
company_apply <- read_excel("20company_final.xlsx")
View(company_apply)
str(company_apply)

# Normalization
company_apply.norm <- company_apply

library(caret)
head(company_apply)
norm.values <- preProcess(company_apply[,c(2:13)], method=c("center","scale"))

company_apply.norm[,c(2:13)] <- predict(norm.values, company_apply[,c(2:13)])

# final*****
library(randomForest)
rf1 <-  randomForest(Y.Bankrupt. ~., data = train.ros.norm,
                     ntree = 3000, mtry= 2, importance = TRUE)
rf1.pred <- predict(rf1, company_apply.norm)
rf1.pred
```

```
> rf1.pred
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Levels: 0 1
```

20개의 국내 기업 모두
파산 위험이 없는 것으로 확인!

05 결론 및 실적용

(3) 후기

어려웠던 점 01

재무 용어 多

-재무 관련 변수명들

-재무 용어가 친숙하지 않아 data 해석의 어려움을 느낌

어려웠던 점 02

다수의 변수

-총 95개의 변수 존재

-예측 변수로 사용될 수 있는 변수가 너무 많아 데이터 처리시 용량 과부화

해결방법

변수들을 카테고리별로 분류하여 이해 노력

변수 축소 과정을 통해 활용되는 변수의 수 감소

수업시간에 배운 내용을 실제 데이터에 적용해보며 데이터 마이닝 기법을 통해 일상생활에서 발생하는 문제를 효율적으로 해결할 수 있음을 느낌

감사합니다

Thank you