Final project

김두은, 김지원, 김태완, 배소연, 편수현, 홍유빈

Part 1,

Preprocessing

Part 1,

Preprocessing

데이터 설명

이 데이터셋은 폴란드 회사의 파산 예측에 관한 것입니다.

이 데이터는 전 세계 신흥 시장에 대한 정보가 포함된 데이터베이스인 Emerging Markets Information Service(EMIS)에서 수집되었으며, 재무제표 상의 각종 수치가 들어있습니다.

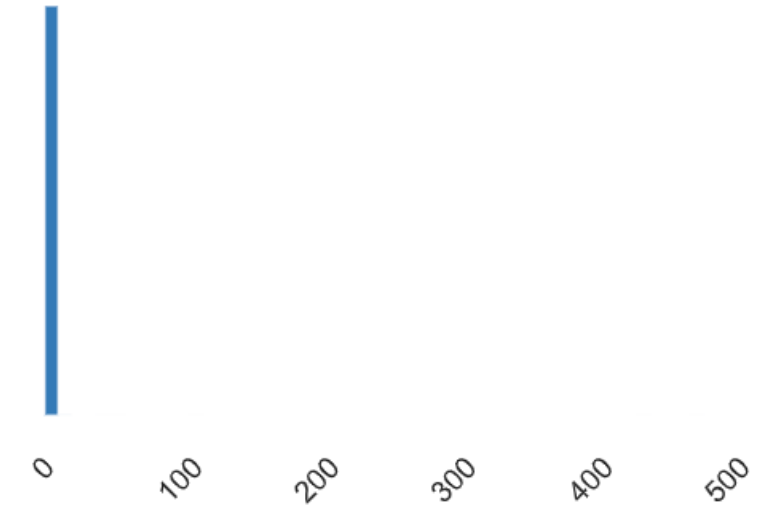
Part 1,

Preprocessing

1. 이상치 제거
2. 변수 간 선형관계 제거
3. 오버 샘플링

Part 1,

Distributions of the data



A

대부분의 변수가 최빈값 근처에 몰려 있되 꼬리가 매우 매우 길게 나온 분포를 보임.

B

따라서 이상치 제거가 꼭 필요.

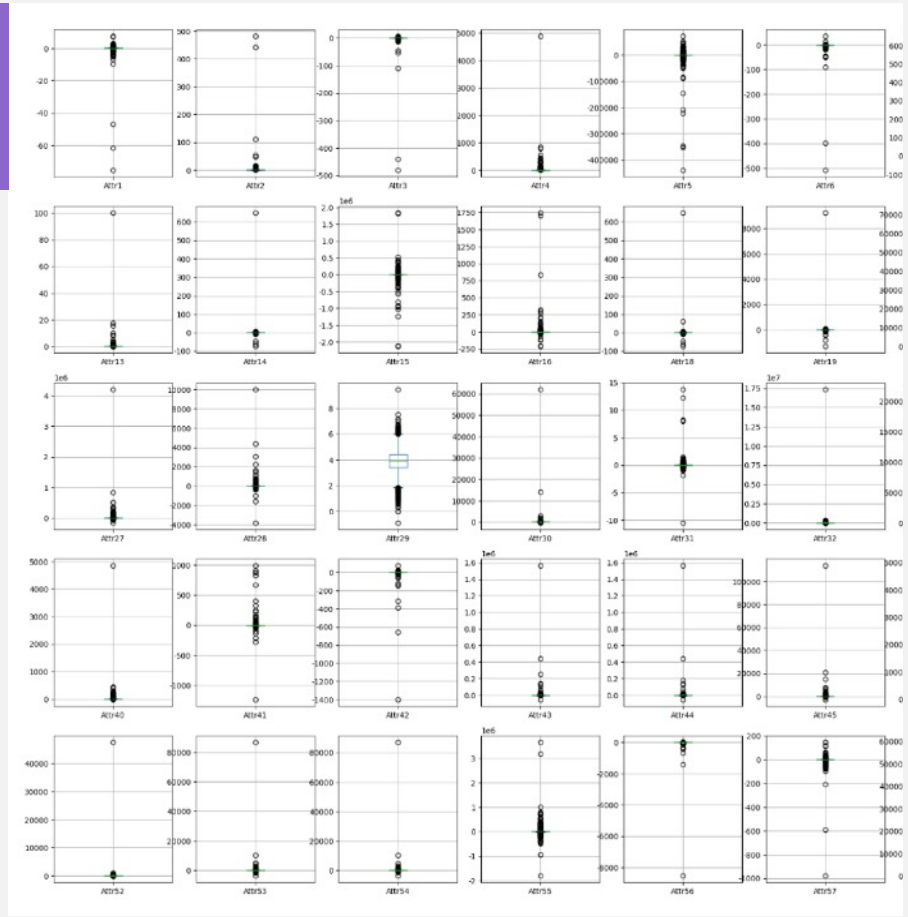
C

다만 데이터가 10,000개로 그리 충분하지 않기 때문에 섣불리 값을 버리면 안 되고, 분포 자체가 원래 넓을 수 있으므로 적당한 정도로만 처리해야 함.

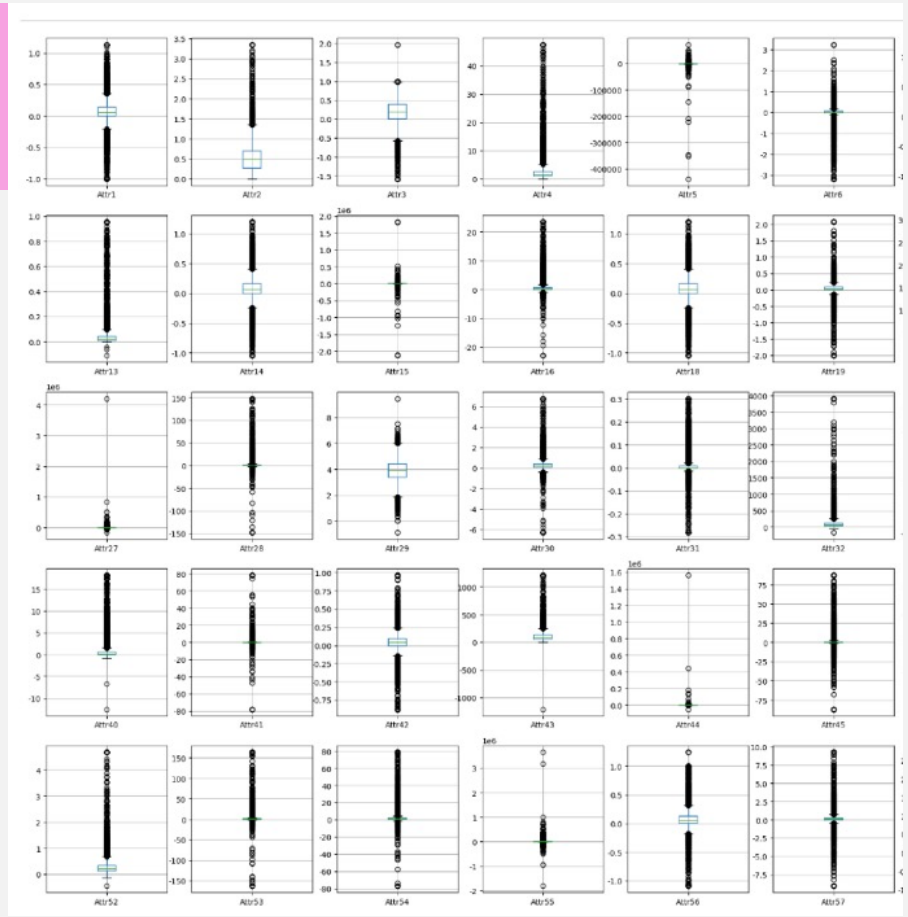
Part 1,

Preprocessing - scaling outliers

A



B



Part 1,

Preprocessing - multicollinearity

STEP 1

$$Attr20 = \frac{365 \times product}{sales}$$

$$Attr60 = \frac{sales}{product}$$

$$Attr13 = \frac{gross\ propit + depreciation}{sales}$$

$$Attr31 = \frac{gross\ propit + interest}{sales}$$

$$Attr19 = \frac{gross\ propit}{sales}$$

$$Attr2 = \frac{total\ liabilities}{total\ assets}$$

$$Attr17 = \frac{total\ assets}{total\ liabilities}$$

>>>

STEP 2

```
from statsmodels.stats.outliers_influence
import variance_inflation_factor
```

Part 1,

Preprocessing – oversampling

STEP 1

```
from sklearn.model_selection
import train_test_split
```

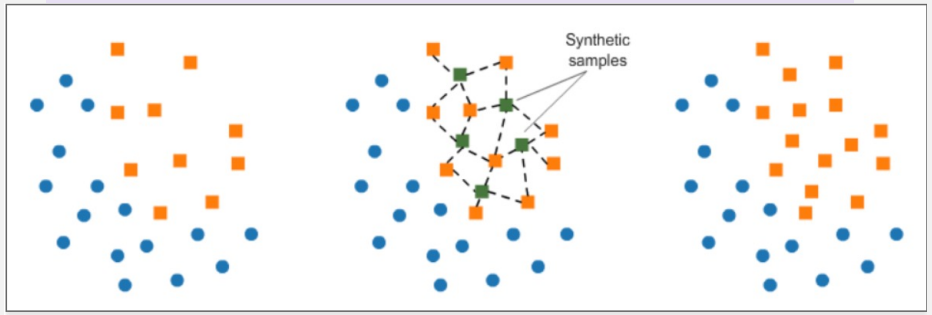
오버 샘플링으로 생성되는 데이터는 가상의 데이터이므로, 트레인 데이터에 대해서만 적용해야 함.

>>>

STEP 2

```
from imblearn.over_sampling import SMOTE
```

8066 >>> 15506



Part 2,

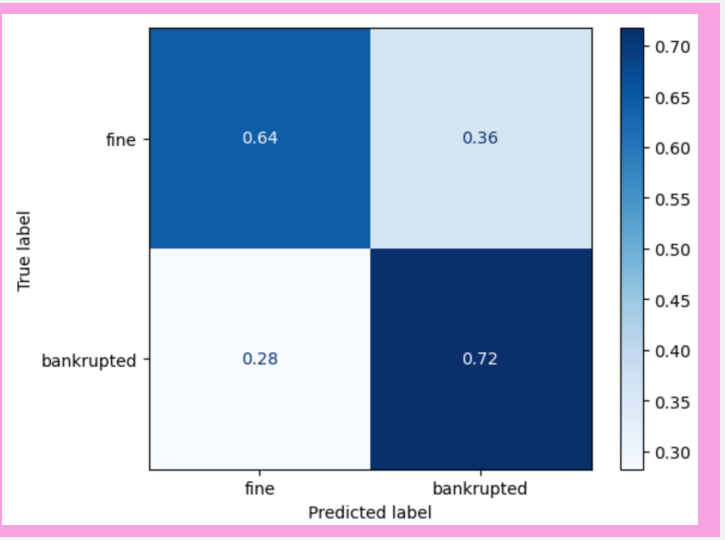
Modeling

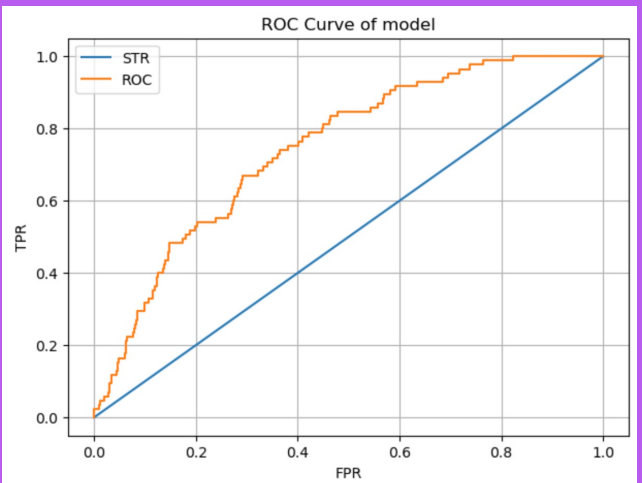
Part 2,

Baseline model - logistic regression

```
from sklearn.linear_model import LogisticRegression
```

```
--- validation result ---
Accuracy(정확도): 0.647
Precision(정밀도): 0.081
Recall(재현율):0.718
F1 Score: 0.146
AUC: 0.681
Confusion Matrix(오차행렬)
: [[1244  688]
 [  24   61]]
```





평가 척도

정확도- 전체 예측 중 맞은 비율 (대각선의 비율)
정밀도- 예측한 positive 중 true positive의 비율
재현율- 실제 positive중 true positive의 비율
F1- 정확도와 정밀도의 조화평균
AUC- roc curve의 면적

혼돈행렬

Label 1을 비교적 잘 예측하지만, 반대급부로 label 0에 대한 오판이 늘어남.

ROC Curve

진양성과 위양성의 상충관계를 나타낸 곡선.
면적이 넓을수록 좋음.

Part 2,

참고용 모델 - Keras simple DNN, ML Models

```
model = Sequential()
model.add(Dense(41, input_shape=(41,), activation='relu'))
model.add(Dense(36, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(36, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
--- validation result ---
Accuracy(정확도): 0.89
Precision(정밀도): 0.152
Recall(재현율):0.353
F1 Score: 0.213
AUC: 0.633
Confusion Matrix(오차행렬)
: [[1765  167]
 [  55   30]]
```

DNN

```
from sklearn.ensemble
import RandomForestClassifier
```

```
RandomForest with Hyperparameter Tuning
Confusion Matrix for test data:
 [[1922   42]
 [  51   20]]
Accuracy score : 0.9543, F1 score : 0.3008
```

Random forest

```
from sklearn.svm import SVC
```

```
SVM with Hyperparameter Tuning
Confusion Matrix for test data:
 [[1884   80]
 [  66    5]]
Accuracy score : 0.9283, F1 score : 0.0641
```

Support vector machine

Part 2,

Bayesian modeling - SSVS

$$Y \sim Bernoulli(\mu(X))$$

$$log\frac{\mu(X)}{1-\mu(X)} = X\beta$$

$$\beta_j \overset{ind}{\sim} \lambda_j N(0,\sigma_1^2) + (1-\lambda_j)N(0,\sigma_2^2)\ for\ j = 1,\cdots,10$$

$$\lambda_j \overset{ind}{\sim} Bernoulli(1/2)$$

$$\sigma_1^2 \sim InverseGamma(1,20),\ \sigma_2^2 \sim Gamma(1,20)$$

Bayesian GLM with Spike and slab prior

1. indicator λ가 1이면 σ_1을 표준편차로 선택 - 매우 큰 분산으로 non-informative prior에 가까움 - beta posterior가 데이터(likelihood)의 영향을 많이 받음.

2. indicator lambda가 0이면 σ_2 를 표준편차로 선택 - 0을 중심으로 한 매우 뾰족한 prior - beta posterior가 0을 벗어나기 힘듦.

3. β, λ, σ_1^2, σ_2^2 의 MCMC 샘플을 생성. 만약 j번째 λ의 평균값이 일정 기준을 넘으면 (inclusion probability > threshold) j번째 변수를 모델에 포함.

Part 2,

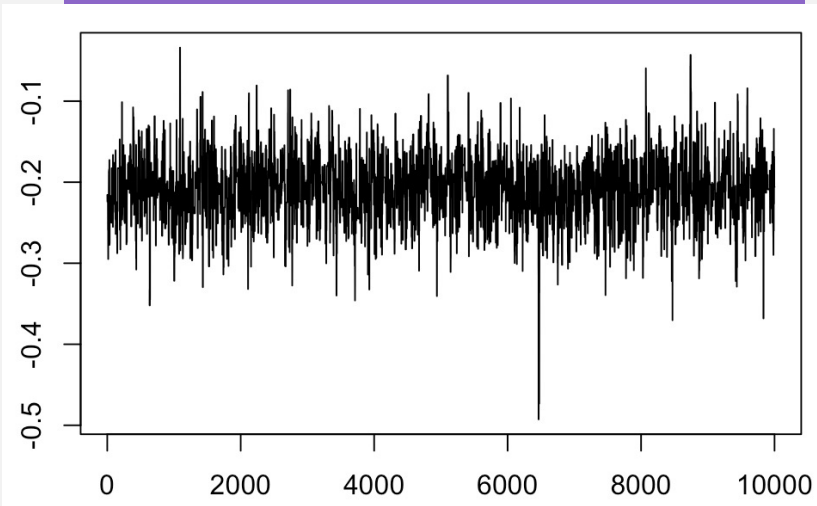
Bayesian modeling - SSVS

```
lambda = matrix(nrow=n, ncol=p)
beta = matrix(nrow=p, ncol=n)
sigma1 = rep(0, n)
sigma2 = rep(0, n)
```

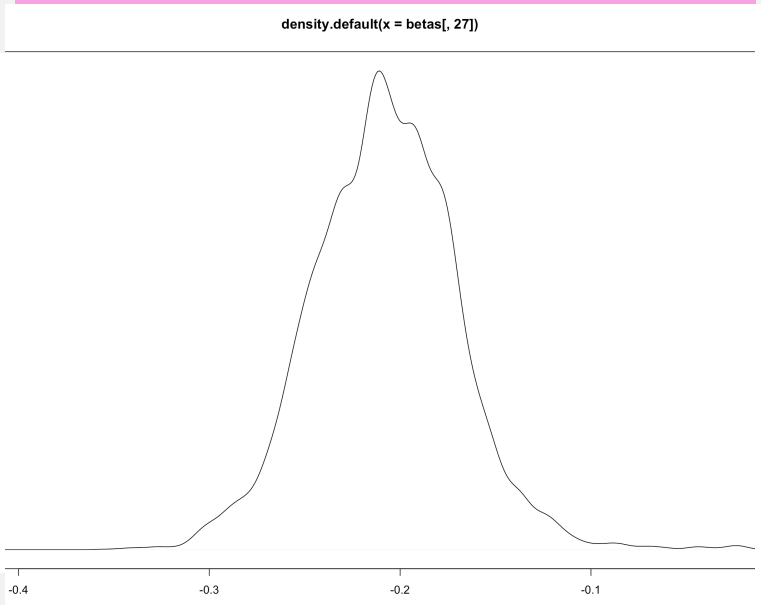
```
#sigma2 block
phi_s2_now = likelihood + dgamma(sigma2[i-1], shape=1, rate=20, log=T)
s2_cand = rnorm(1, mean=sigma2[i-1], sd=0.05)
if(s2_cand < 0){sigma2[i] = sigma2[i-1]}else{
  phi_s2_cand = likelihood + dgamma(s2_cand, shape=1, rate=20, log=T)
  if((phi_s2_cand - phi_s2_now) > log(runif(1))){
    sigma2[i] = s2_cand;
  }else{
    sigma2[i] = sigma2[i-1]
  }
}
```

Part 2,

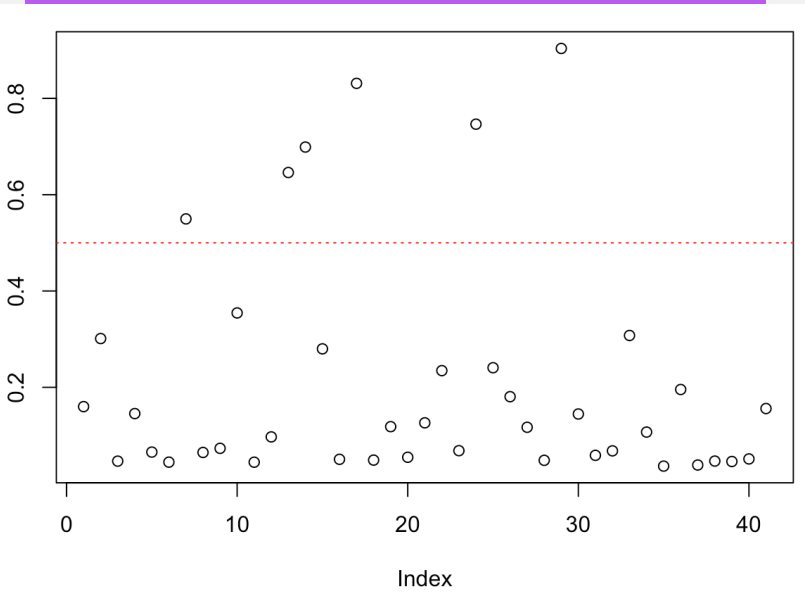
Bayesian modeling - SSVS



Trace plot



Posterior density



Inclusion probabilities

Part 2,

Bayesian modeling - SSVS

```
          Reference
Prediction    0    1
         0 1248   22
         1  684   63
```

```
> Acc
[1] 0.6499752
> Precision
[1] 0.08433735
> Recall
[1] 0.7411765
> f1
[1] 0.1514423
```

evaluation

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4983 2586
         1 2770 5167

               Accuracy : 0.6546
                 95% CI : (0.647, 0.6621)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16
```

Train score

Posterior mean을 point estimator로 사용했을 때, 최소한 기존 모델보다 성능이 떨어지지 않음 →

변수 개수를 많이 줄였으므로 기존 모델에 비해 설명에 유리함.
그러나 ML, DL 모델에 비해 성능이 좋지 않다는 한계가 있음.

Part 2,

Posterior predictive check

```
#posterior predictive check
selected = colnames(X)[which(apply(lambda,2,mean)>0.3)]

Xsel = subset(train_x, select=selected)
Xsel = as.matrix(Xsel)
betasel = beta[which(apply(lambda,2,mean)>0.3), 1001:11000]
yselp = invlogit(Xsel %*% betasel)
ysel=matrix(nrow=15506, ncol=10000)
for(i in 1:10000){
  for(j in 1:15506){
    ysel[j,i] = rbinom(1, size=1, prob=yselp[j,i])
  }
}
```

```
> for(i in 1:10000){ssum[i] = ifelse(mean(yseli[,i]) > 0.5, 1, 0)}
> sum(ssum)/10000
[1] 0.4299
```

Posterior mean을 point estimator로 사용하면 β 자체의 불확실성을 반영할 수 없음 →

β의 MCMC 샘플에서 y의 posterior draw를 추출한 뒤 추출된 y가 원래 데이터 y_0의 분포를 잘 반영하는지 확인

≡

통계량 $d(y)$가 원래 데이터의 통계량 $d_0(y)$를 중심으로 분포되어 있는지 확인.

$$Bayesian\ p\ value = \frac{1}{T}\sum_{t=1}^{T} I(d_t > d_0)$$

감사합니다