2021-1 YDMS 2주차 과제

〈Subject : UNBALANCE DATA 처리 및 모델 성과평가〉

김하은 2019251034

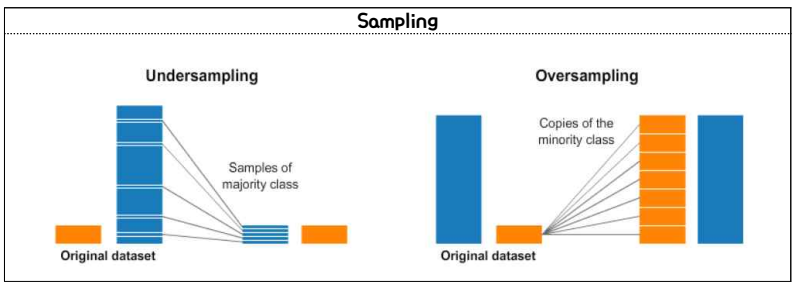
1. 계급 불균형 자료(Imbalanced data) 및 해결법에 대하여 조사

계급 불균형 자료란, 범주형 계열인 Target 변수의 계급 균형이 심하게 불균형을 이룬 자료이다. 따라서 범주의 비율이 균일하지 않고 한쪽으로 치우쳐져 있다. 가령 의료관련 자료에서 암환자와 정상인을 구분할 때 암환자의 수가 적기 때문에 '암환자' 와 '정상인'의 균형비가 매우 불균형하게 나타난다. 이러한 자료를 계급 불균형 자료라 한다. 계급 불균형 자료에서 상대적으로 다수를 차지하는 class는 'major class'라 하고, 상대적으로 소수를 차지하는 class를 'minor class'라 한다. (물론 계급비가 2:8, 1:99 와 같은 경우를 계급 불균형 자료라 일컷는다.)

두 계급의 관측 개체수가 심한 불균형을 이루는 경우 소수 계급(minor class)을 다수 계급(major class)으로 오분류 하는 경향이 있다. 하지만 소수 계급에 더 관심 있는 분석의 경우 앞서 언급한 오분류는 분석 예측성능을 하향시킨다. 대부분 실존하는 데이터에서 우리가 관심을 가지는 경우는 소수 계급이므로, 계급 불균형에 대한 해결이 필요하다.

계급 불균형 자료의 해결법은 크게 Sampling (over, under, over&under) 기법과 SMOTE (instance 이용) 기법이 있다.

①Sampling (over, under, over&under) 기법



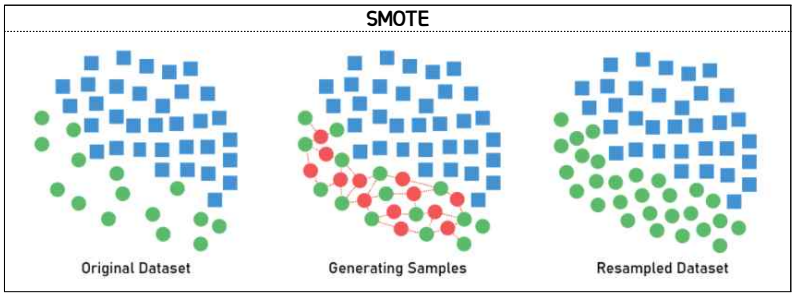
Sampling 기법은 계급의 개수를 맞추어주는 방법으로 다수 집단을 소수 집단에 맞추거나, 소수 집단을 다수 집단에 맞추는 방법이 있다. 먼저 Under Sampling (Down Sampling)은 다수 집단을 비복원 랜덤 추출을 이용해 소수 집단에 맞추는 방법이다. 이 방법은 Sampling 시간이 오래 걸리지 않고 실제 데이터에서 뽑은 데이터로 모델을 훈련시키기 때문에 편향이 덜 발생한다는 장점이 있다. 그러나 실제 데이터에서 추출되지 않은 다수의 데이터들이 손실되기 때문에 예측력이 떨어질 수 있는 위험성을 가지고 있다.

다음으로 Over Sampling은 소수 집단을 복원 랜덤 추출하여 다수 집단에 맞추는 방법이다. 이 방법은 기존의 소수 계급에서 데이터를 복원 추출하기 때문에 Under Sampling과 달리 정보의 손실이 최소화 된다는 장점이 있다. 하지만 반대로 데이터가 중복으로 사용되는 것이기 때문에 예측 편향이 발생하기 쉽다. 예측 편향이 발생한다는 것은 과적합의 위험성이 크다는 것으로 일반화 되지

못한 특정 모델이 만들어지기 쉽다는 것이다.

앞서 소개한 두 가지 Sampling에서 발생한 정보의 손실로 인한 예측력 저하 혹은 중복 데이터 사용으로 인한 과적합의 우려를 줄이기 위해 고안해낸 Sampling 기법이 바로 Under&Over Sampling 기법이다. 소수 계급의 일부분을 Over Sampling하고 다수 계급의 일부분을 Under Sampling 하여 계급의 균형을 맞추어주는 방법이다. 이러한 방법은 기존의 단순 Sampling기법에서 발생한 문제점을 어느 정도 보완한다는 장점이 있다. 정보의 손실을 줄이고 동시에 일정 수준 이상의 편향이 발생하지 않도록 하는 방법이기 때문이다.

②SMOTE 기법



위의 그림은 SMOTE 기법을 잘 표현해낸 그림이라 생각되어 첨부하였다. SMOTE 기법은 소수 계급의 데이터들 중 랜덤으로 선택한 하나의 데이터와 주변에 유클리드 거리가 가까운 데이터 점을 선택하여 가상의 선을 만들고 그 위에 임의의 점을 선택하여 이를 새로운 Sample로 추가하는 방법이다. 위의 그림에 연결하여 설명하면 더 쉽게 이해가 가능하다. 다수 계급은 파란색 네모 집단이고 소수 계급은 초록색 동그라미 집단이다. 계급의 불균형을 해결하기 위해, 소수 계급인 초록색 동그라미들 중 하나를 선택하고, 이와 유클리드 거리가 가까운 점 (빨간색 표시)을 선택하여 임의의 점선을 그린다. 그 점선 위에 랜덤하게 점을 선택하여 새로운 Sample 로 추가하는 것이 SMOTE 기법이다.

이 방법은 랜덤 복원 추출하는 Over Sampling보다 과적합이 일어날 가능성이 낮으며, 추가 Sample을 만드는 것이기 때문에 Under Sampling와는 달리 정보 손실의 우려가 없다는 장점이 있다. 하지만 소수 계급의 유클리드 거리만을 고려하기 때문에 다수 계급의 위치는 고려되지 않고 이로 인해 계급이 겹치거나 오차를 만들 수 있는 우려가 있다. 또한 유클리드 거리로 추가 Sample을 생성하는 것이므로 고차원 데이터 셋에서는 효율적이 않다는 단점이 있다.

2. 정확도의 역설(Accuracy Paradox)에 대하여 조사

		실제	
		positive	negative
예측	positive	TP	FP
	negative	FN	TN

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

위의 표와 수식은 '정확도(*Accuracy*)' 에 관한 내용이다. 결과적으로 정확도는 맞은 것을 맞다고,

틀린 것을 틀리다고 예측해야 정확도가 올라가고, 예측 성능이 올라갈 것이다. 이러한 일반적인 생각에 반하는 것이 '정확도의 역설'이다. 자료에 Negative가 많은 경우, 즉 틀린 것을 틀리다고 예측하는 경우 정확도의 역설이 발생하기 쉽다.

가령 '암 진단'에 관한 데이터를 생각해보면, 암환자의 데이터는 정상인의 데이터보다 극히 적다. 특히 (Target) 변수의 비율이 2:8, 1:9 와 같이 심한 불균형을 이루는 경우, 변수의 분포는 희귀한 사건에 집중하지 않는 편향성을 보인다. 이러한 경우 정작 제대로 분류 해야하는 희귀 사건(암환자 예측)은 분류가 되지 않고 빈도가 극도로 큰 사건 (정상인 예측)에 비중이 쏠리게 된다. 그럼에도 불구하고 분류모형은 입력변수와는 관계없이 모두 정확도가 매우 높게 측정된다. 하지만 정확도가 높음에도 이 모델은 무의미한 모델이다. 실제 예측해야하는 값은 '암환자를 암환자라고 예측하는 것'이지 '정상인을 정상인으로 예측하는 것'이 아니다. 마치 한여름에 눈이 오지 않을 확률을 구하는 것과 같다. 이와 같은 것을 '정확도의 역설'이라고 칭한다.

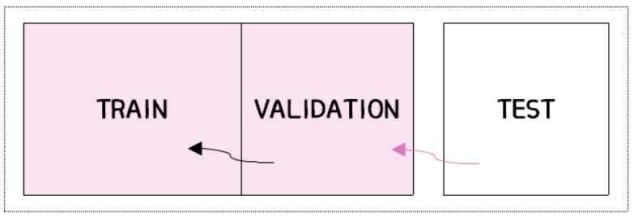
따라서 계급 불균형 자료를 이용한 모델의 경우 정확도에만 치우쳐진 모델 성능평가를 해서는 안된다. ROC 그래프나 특이도, 민감도를 모두 고려하여 모델의 성능평가를 해야 가장 효율적이 모델을 구축할 수 있다.

3. 데이터 셋을 TRAIN, VALIDATION, TEST 로 나누는 이유에 대하여 조사

Train dataset이란 모델을 학습하기 위한 dataset이다. 오로지 학습만을 위한 데이터 셋이라고 생각하면 된다. Validation dataset이란 학습이 완료된 모델을 검증하기 위한 데이터셋이다. 즉 Train과정을 거친 모델에 검증용 데이터인 Validation을 넣어 검증하는 것이다. 이 과정에서 과적합이나 예측율을 미리 보고 모델을 수정하거나 변경할 수 있다. 학습과 검증이 끝나면 마지막으로 최종 검증을 하게 되는데 이때 사용하는 것이 바로 Test dataset이다.

정리하자면, 모델을 구축시키는 영역에 Train dataset과 Validation dataset이 존재하는 것이고, 모델의 최종 검증에 해당하는 영역에 Test dataset가 존재하는 것이다. 이를 크게 TRAIN, TEST로 나누어 생각하는데, 보통 데이터를 8:2로 나누는 것이 일반적이다. 여기에서 다시 TRAIN 영역을 6:2로 나누어 결국 Train : Validation : Test는 6:2:2로 나누어진다.

위에서 설명한 것을 바탕으로 그림을 그려보면 다음과 같다. Train은 모델을 학습시키는 과정이고 이를 검증하는 것이 Validation이다. 이때 Validation은 모델을 학습시키진 않지만 모델 수정이나 최종 모델 설정에 관여를 한다. 학습과 검증이 끝나면 최종적으로 Test를 이용하여 모델을 검증하게 된다.

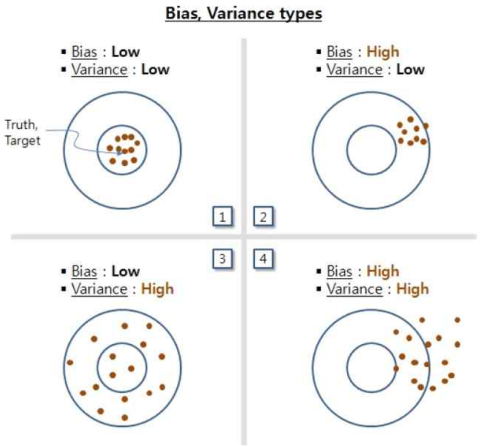


그런데 VALIDATION과 TEST는 검증한다는 점에서 비슷한 역할을 수행하게 되는데 Validation

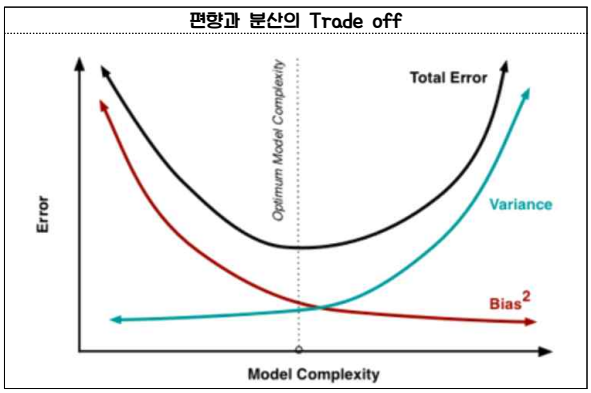
이 글이 왜 필요한지 의문이 들었고 이에 대해 공부를 해보았다. 이에 대한 답은 다음과 같다.

TEST가 있는데 굳이 VALIDATION이 필요한가?
: 학습 단계에서 모델의 수정이나 변경을 가능하게 하는 것이 VALIDATION의 역할이다. 학습단계에서 TEST 데이터 셋을 이용하게 되면 최종적으로 선정한 모델을 검증할 수 없게 되므로, TEST 데이터 셋은 최종 검증 이전까지 절대 건들어서는 안된다.

4. 과적합(과대적합)에 대하여 조사

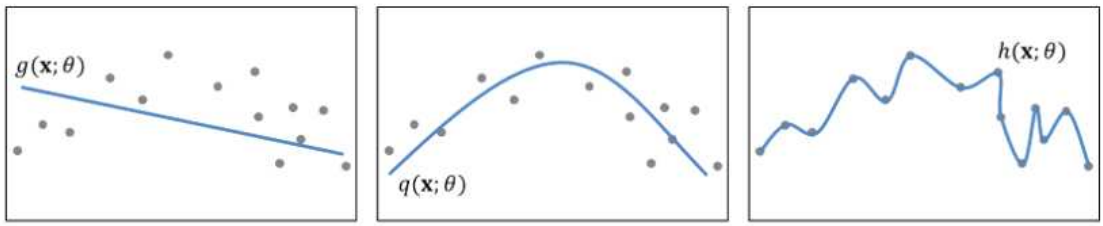


과적합을 알아보기 앞서 편향과 분산에 대해 알아보았다. 위의 그림은 편향과 분산을 잘 표현해낸 그림이라 생각하여 첨부하였다. 편향은 Target과 평균이 떨어진 척도이고, 분산은 데이터들의 흩어진 정도를 편차를 이용하여 나타낸 값이다. 편향과 분산이 작을수록 가장 좋은 모델이라 할 수 있다. 하지만 편향과 분산은 서로 Tarde off라는 난제를 가지고 있다. 한 쪽이 작아지면 다른 한 쪽이 커진다. 즉 분산이 작을수록 편향이 커지고, 분산이 클수록 편향이 작아진다. 이는 아래의 그림과 같다.



TOTAL ERROR는 편향이나 분산 둘 중 하나라도 올라가게 되면 같이 커지는데, 분산이 커지면 Overfitting (과적합)이 일어나고, 편향이 커지면 Underfitting (과소적합)이 일어나게 된다.

그렇다면 과소적합과 과적합은 무엇인가? 이는 다음의 그래프를 보면 이해하기 쉽다.



(a) Underfit (b) Ideal fit (c) Overfit

과소적합(underfitting)이란 데이터를 설명하기 부족한 모델이며 (a)그래프가 해당된다. 반면 과적합(overfitting)이란 너무 학습이 많이 되어 일반화 되지 못한 모델이며 (C)그래프가 해당된다. 과적합이 일어나는 경우 해당 TRAIN에는 잘 맞는 그래프이지만 다른 값을 넣게 되면 그래프가 제대로 예측할 수 없다. 즉 TRAIN에만 적합한 모델이므로 좋지 않은 모델이다. 주로 모델의 학습이 많이 된 경우 과적합이 발생하게 되는데, 이는 많은 Validation을 거칠수록 해결이 가능하다. 이와 관한 분석법에는 대표적으로 K-fold가 있다.

5. 분류모형의 예측 성능부분에서 사용할 수 있는 적합 통계량들에 대해서 조사

분류모형의 예측 성능을 평가할 때, 정확도만을 고려하면 앞서 언급한 '정확도의 역설'과 같은 문제점이 발생한다. 따라서 정확도와 더불어 민감도(sensitivity) 와 특이도(specificity)를 사용하여 성능을 평가한다. 이를 바탕으로 다음과 같은 3가지 통계량들을 살필 수 있다.

①민감도와 특이도
민감도란 1인 경우 1이라 예측한 경우이고, 특이도란 0인 경우 0이라 예측한 것이다.

②TPR과 FPR
TPR이란 양성율로, 민감도와 같은 개념이다. 즉 1인 경우를 1이라 예측한 경우이다.
가령 '암 진단 데이터'에서 암환자에게 암을 진단한 경우가 해단된다.
FPR이란 위양성율로 (1-특이도)이다. 즉 0인 경우 1로 잘못 예측한 경우이다.
가령 '암 진단 데이터'에서 정상인에게 암을 진단한 경우가 해당된다.

③ROC곡선에서 얻은 AUC(곡선 아래의 영역의 넓이)
TPR과 FPR은 서로 반비례 관계를 보인다. 이를 그래프로 표현한 것이 바로 ROC커브이다. ROC커브의 그래프 밑 면적이 1에 가까워질수록 성능이 좋은 것이며, 0.50이면 성능이 전혀 없는 것으로 판단된다.

6. DISCUSSION 질문 정해오기 (과제에 첨부)

Validation 이 중요한 과정이라 배웠는데, 분석은 TRAIN과 TEST로 진행하는 이유

✓ Unbalanced Data 처리 실습

7. 해당 데이터에 대해서 계급 불균형 자료에 대한 해결을 적용 및 해석

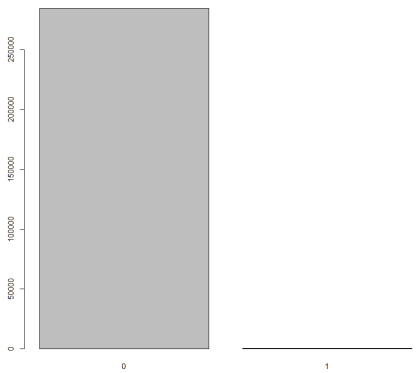
Ⅰ.EDA | Ⅱ.데이터 나누기 | Ⅲ.샘플링/SMOTE | Ⅳ.모델 학습 | Ⅴ.모델 성능평가

Ⅰ.EDA 진행

이번 데이터에 대한 요약본은 다음과 같다. 총 31개의 변수로 이루어져 있으며 284,807 개의 데이터들로 구성되어있다. Class 데이터를 제외한 나머지 변수들은 모두 연속형 변수이며, 범주형 변수인 Class 데이터의 경우 심각한 계급 불균형을 이루고 있음을 시각화를 통해 확인하였다. 또한 연속형 변수들의 경우 상관관계가 크지 않음을 알 수 있다.

```
> summary(credit)
      Time                V1                  V2                  V3                  V4                  V5                  V6
 Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605
 1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683
 Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799   Median :-0.01985   Median :  -0.05434   Median : -0.2742
 Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000
 3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986
 Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016
       V7                  V8                  V9                 V10                 V11                 V12                 V13
 Min.   :-43.5572   Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747   Min.   :-18.6837   Min.   :-5.79188
 1st Qu.: -0.5541   1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249   1st Qu.: -0.4056   1st Qu.:-0.64854
 Median :  0.0401   Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276   Median :  0.1400   Median :-0.01357
 Mean   :  0.0000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000
 3rd Qu.:  0.5704   3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959   3rd Qu.:  0.6182   3rd Qu.: 0.66251
 Max.   :120.5895   Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891   Max.   :  7.8484   Max.   : 7.12688
       V14                 V15                 V16                 V17                 V18                 V19                 V20
 Min.   :-19.2143   Min.   :-4.49894   Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746   Min.   :-7.213527   Min.   :-54.49772
 1st Qu.: -0.4256   1st Qu.:-0.58288   1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850   1st Qu.:-0.456299   1st Qu.: -0.21172
 Median :  0.0506   Median : 0.04807   Median :  0.06641   Median : -0.06568   Median :-0.003636   Median : 0.003735   Median : -0.06248
 Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000   Mean   : 0.000000   Mean   :  0.00000
 3rd Qu.:  0.4931   3rd Qu.: 0.64882   3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807   3rd Qu.: 0.458949   3rd Qu.:  0.13304
 Max.   : 10.5268   Max.   : 8.87774   Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069   Max.   : 5.591971   Max.   : 39.42090
       V21                 V22                 V23                 V24                 V25                 V26                 V27
 Min.   :-34.83038   Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663   Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679
 1st Qu.: -0.22839   1st Qu.: -0.542350   1st Qu.: -0.16185   1st Qu.:-0.35459   1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840
 Median : -0.02945   Median :  0.006782   Median : -0.01119   Median : 0.04098   Median :  0.01659   Median :-0.05214   Median :  0.001342
 Mean   :  0.00000   Mean   :  0.000000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000
 3rd Qu.:  0.18638   3rd Qu.:  0.528554   3rd Qu.:  0.14764   3rd Qu.: 0.43953   3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045
 Max.   : 27.20284   Max.   : 10.503090   Max.   : 22.52841   Max.   : 4.58455   Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198
       V28                Amount              Class
 Min.   :-15.43008   Min.   :    0.00   Min.   :0.000000
 1st Qu.: -0.05296   1st Qu.:    5.60   1st Qu.:0.000000
 Median :  0.01124   Median :   22.00   Median :0.000000
 Mean   :  0.00000   Mean   :   88.35   Mean   :0.001728
 3rd Qu.:  0.07828   3rd Qu.:   77.17   3rd Qu.:0.000000
 Max.   : 33.84781   Max.   :25691.16   Max.   :1.000000
```

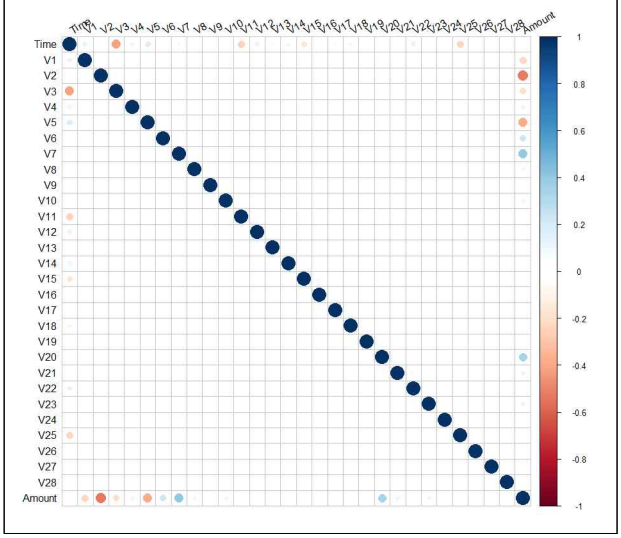
UNBALANCE DATA of Class (Target 변수)




```
> table(credit$Class)

     0      1
284315    492
```

연속형 변수 간 상관관계



*컴퓨터 오류로 범주형 변수(타겟 변수)와의 연관성 시각화는 진행하지 못했습니다.

II. 데이터 랜덤 나누기

다음은 총 31개의 변수의 284,807개의 데이터들을 Train data set, Test data set 로 나눈 것이다. 데이터는 7:3 으로 랜덤하게 나누었으며, Train은 199,487개, Test는 85,320개로 나누어짐을 알 수 있다.

```
> #데이터 분류
> set.seed(1234)
> idx <- sample(x = c("train", "test"),
+               size = nrow(credit),
+               replace = TRUE,
+               prob = c(7, 3))
>
> train<-credit[idx == "train", ]
> test<-credit[idx == "test", ]
> dim(train)
[1] 199487     31
> dim(test)
[1] 85320    31
```

분석에 들어가기 앞서 타겟 변수인 Class를 factor요인으로 변경해주었다. 이 과정이 생략되면 범주형 변수의 모델을 구축할 수 없기 때문에 반드시 필요한 과정이다.

```
as.factor(train$Class)->train$Class
as.factor(test$Class)->test$Class
str(test)
str(train)
```

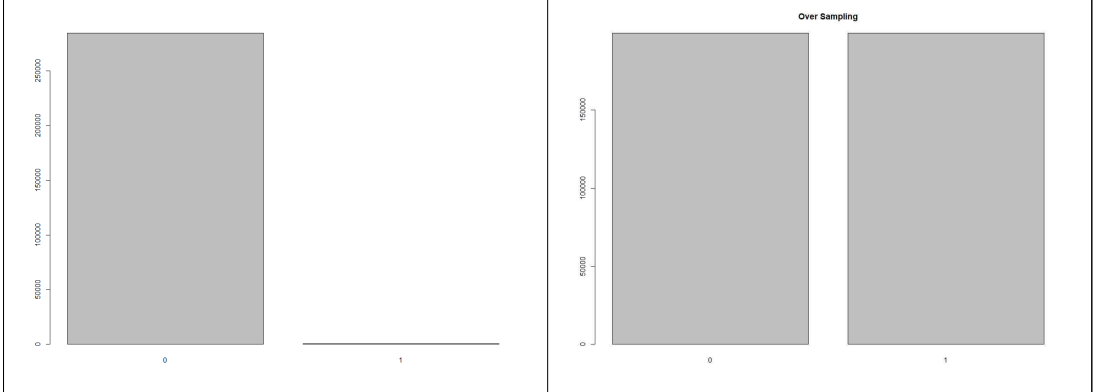
```
$ Class  : int    0 0 0 0 0 0 0 0 0 0 ...
$ Class : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

III. 샘플링/SMOTE

다음은 계급 불균형 자료를 Sampling 기법 혹은 SMOTE 기법을 활용하여 계급의 균형을 맞추어주는 작업이다. 각 4개의 경우로 나누어 진행하였으며 순서대로 Over Sampling, Under Sampling, Under & Over Sampling, SMOTE 기법이다.

Sampling을 진행할 때 주의해야하는 점은, Train Dataset만을 이용해야 한다는 점이다. 이를 주의하여 Sampling을 하였고, 이를 아무 처리하지 않은 기존의 Data와 비교해보았다.

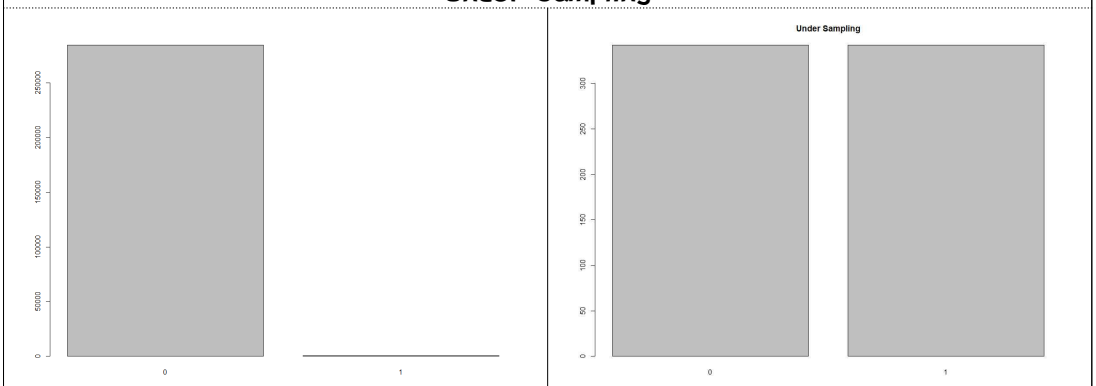
Over Sampling




```
> data_balanced_over<-ovun.sample(Class~.,data=train,method="over",N=398290)$data
> table(data_balanced_over$Class)

     0      1
199145 199145
```

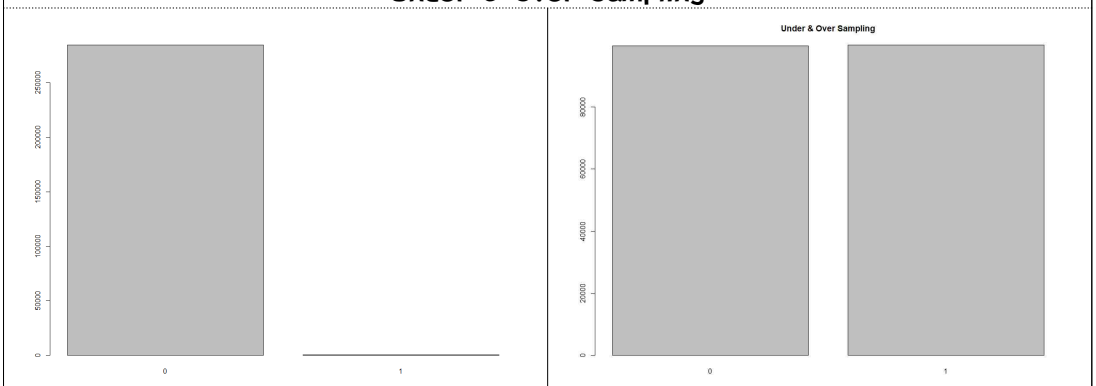
Under Sampling




```
> data_balanced_under<-ovun.sample(Class~.,data=train,method="under",N=684)$data
> table(data_balanced_under$Class)

  0   1
342 342
```

Under & Over Sampling




```
> data_balanced_both <- ovun.sample(Class ~ ., data = train, method = "both", p=0.5, N=199487, seed = 1)$data
> table(data_balanced_both$Class)

    0     1
99591 99896
```

SMOTE

```
> data.rose <- ROSE(Class ~ ., data = train, seed = 1)$data
```

Ⅳ. 모델 학습

위에서 나눈 모델을 학습시키는 작업이다. 의사결정나무로 모델을 돌렸으며 아래와 같이 진행하였다.

모델 생성

```
tree.rose <- rpart(Class~.,data=data.rose)
tree.under <- rpart(Class~.,data=data_balanced_under)
tree.over <- rpart(Class~.,data=data_balanced_over)
tree.both <- rpart(Class ~ ., data = data_balanced_both)
```

모델 설정

```
pred.tree.rose <- predict(tree.rose, newdata = test)
pred.tree.under <- predict(tree.under, newdata = test)
pred.tree.over <- predict(tree.over, newdata = test)
pred.tree.both <- predict(tree.both, newdata = test)
```

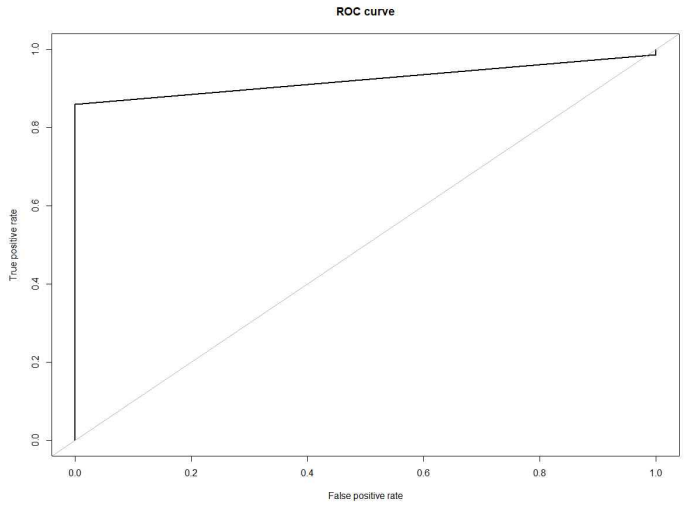
V. 모델 성능평가

다음은 각 모델에 대한 성능을 평가한 것이다. 계급 불균형을 처리하지 않은 모델부터 각 Sampling, SMOTE 기법으로 균형을 맞춘 모델까지 성능을 평가해보았다. 모델 성능은 ROC곡선을 이용하여 판단하였으며 그 중에서도 곡선 밑면의 넓이에 해당하는 AUC로 최종 판단하였다.

예상과 같이 아무 처리를 하지 않은 계급 불균형 모델이 0.923으로 성능이 가장 낮았다. 성능이 가장 높게 나온 것은 Under Sampling 으로, AUC값이 0.963 이 나왔다. 나머지 Over Sampling 과 Under & Over Sampling은 AUC값이 0.958로 동일했으며, SMOTE 의 경우 AUC값이 0.946으로 가장 낮았다. 하지만 이 역시 아무 처리 하지 않은 불균형 모델보다 성능이 높게 평가되었다.

결과적으로, 주어진 계급 불균형 자료(creditcard) 의 경우 Under Sampling을 했을 때가 의사결정나무의 모델에서 가장 성능이 좋게 평가됨을 알 수 있다.

아무 작업하지 않은 DATA



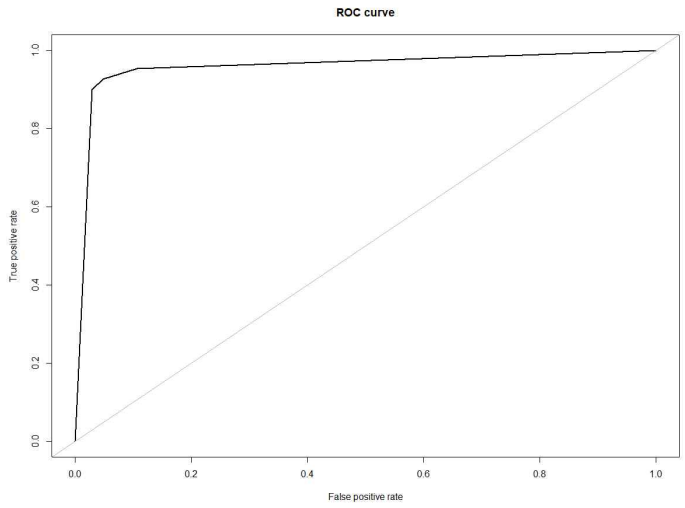

```
> accuracy.meas(test$Class,pred.treeimb[,2])

Call:
accuracy.meas(response = test$Class, predicted = pred.treeimb[,
    2])

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.892
recall: 0.827
F: 0.429
> roc.curve(test$Class, pred.treeimb[,2], plotit=T)
Area under the curve (AUC): 0.923
```

Over Sampling



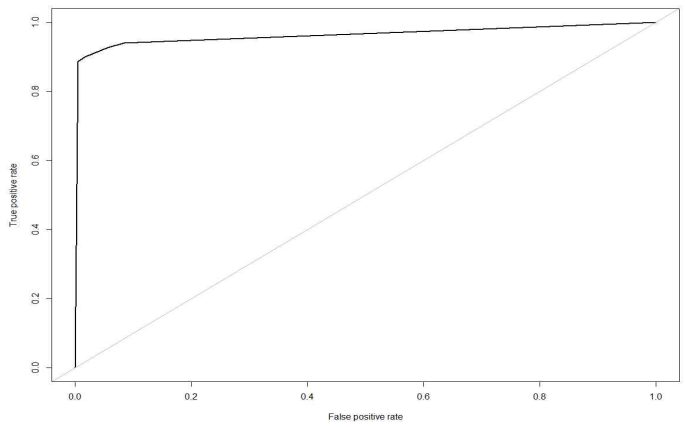

```
> accuracy.meas(test$Class,pred.tree.over[,2])

Call:
accuracy.meas(response = test$Class, predicted = pred.tree.over[,
    2])

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.033
recall: 0.927
F: 0.032
> roc.curve(test$Class,pred.tree.over[,2])
Area under the curve (AUC): 0.958
```

Under Sampling



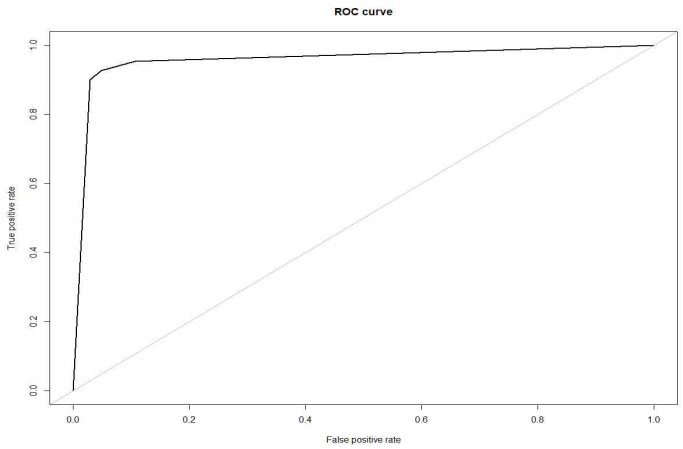

```
> accuracy.meas(test$Class,pred.tree.under[,2])

Call:
accuracy.meas(response = test$Class, predicted = pred.tree.under[,
    2])

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.086
recall: 0.900
F: 0.078
> roc.curve(test$Class,pred.tree.under[,2])
Area under the curve (AUC): 0.963
```

Under & Over Sampling



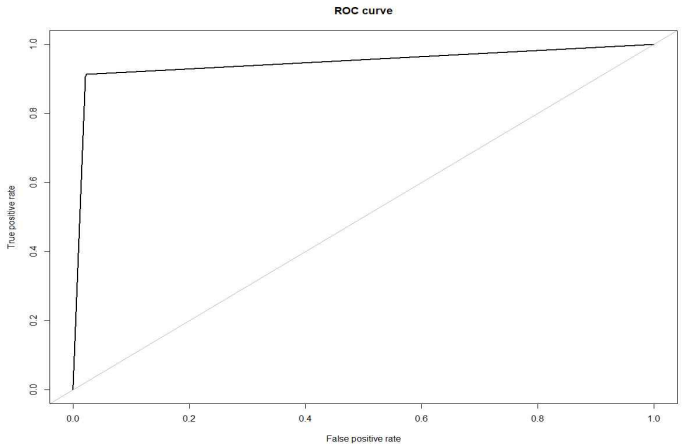

```
> accuracy.meas(test$Class,pred.tree.both[,2])

Call:
accuracy.meas(response = test$Class, predicted = pred.tree.both[,
    2])

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.033
recall: 0.927
F: 0.032
> roc.curve(test$Class,pred.tree.both[,2])
Area under the curve (AUC): 0.958
```

SMOTE




```
> accuracy.meas(test$Class,pred.tree.rose[,2])

Call:
accuracy.meas(response = test$Class, predicted = pred.tree.rose[,
    2])

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.062
recall: 0.913
F: 0.058
> roc.curve(test$Class,pred.tree.rose[,2])
Area under the curve (AUC): 0.946
```