2022 KUBIG 머신러닝 분반 금융팀 기업 파산 예측

16기 민윤기, 박종혁, 임채명, 최규빈

목차

Raw Data Examination

Data Preprocessing

Modeling

필요한 라이브러리 호출


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly.subplots import make_subplots
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score, accuracy_score, classification_report
from sklearn import preprocessing
from imblearn.over_sampling import SMOTE
from sklearn.metrics import roc_auc_score
from sklearn.calibration import CalibratedClassifierCV, calibration_curve
from sklearn.isotonic import IsotonicRegression
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```python
data = pd.read_csv('/content/drive/MyDrive/머신러닝 스터디/Bankrupcy 프로젝트/data.csv')
```

데이터 살펴보기

Raw Data Examination

```
1 data.head()
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Income to Total Assets	Total assets to GNP price	No-credit Interval	Gross Profit to Sales	Net Income to Stockholder Equ
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.622879	0.601453	0.8278
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.623652	0.610237	0.8399
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.623841	0.601449	0.8367
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.622929	0.583538	0.8346
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.623521	0.598782	0.8399

5 rows × 96 columns

변수 살펴보기

Y - Bankrupt?: Class label
X1 - ROA(C) before interest and depreciation before interest: Return On Total Assets(C)
X2 - ROA(A) before interest and % after tax: Return On Total Assets(A)
X3 - ROA(B) before interest and depreciation after tax: Return On Total Assets(B)
X4 - Operating Gross Margin: Gross Profit/Net Sales
X5 - Realized Sales Gross Margin: Realized Gross Profit/Net Sales
X6 - Operating Profit Rate: Operating Income/Net Sales
X7 - Pre-tax net Interest Rate: Pre-Tax Income/Net Sales
X8 - After-tax net Interest Rate: Net Income/Net Sales
X9 - Non-industry income and expenditure/revenue: Net Non-operating Income Ratio
X10 - Continuous interest rate (after tax): Net Income-Exclude Disposal Gain or Loss/Net Sales
X11 - Operating Expense Rate: Operating Expenses/Net Sales
X12 - Research and development expense rate: (Research and Development Expenses)/Net Sales
X13 - Cash flow rate: Cash Flow from Operating/Current Liabilities
X14 - Interest-bearing debt interest rate: Interest-bearing Debt/Equity
X15 - Tax rate (A): Effective Tax Rate
X16 - Net Value Per Share (B): Book Value Per Share(B)
X17 - Net Value Per Share (A): Book Value Per Share(A)
X18 - Net Value Per Share (C): Book Value Per Share(C)
X19 - Persistent EPS in the Last Four Seasons: EPS-Net Income
X20 - Cash Flow Per Share
X21 - Revenue Per Share (Yuan ¥): Sales Per Share
X22 - Operating Profit Per Share (Yuan ¥): Operating Income Per Share
X23 - Per Share Net profit before tax (Yuan ¥): Pretax Income Per Share
X24 - Realized Sales Gross Profit Growth Rate
X25 - Operating Profit Growth Rate: Operating Income Growth
X26 - After-tax Net Profit Growth Rate: Net Income Growth
X27 - Regular Net Profit Growth Rate: Continuing Operating Income after Tax Growth
X28 - Continuous Net Profit Growth Rate: Net Income-Excluding Disposal Gain or Loss Growth
X29 - Total Asset Growth Rate: Total Asset Growth

비슷한 값을 포함하는 변수가 많다

=> 변수 선택 과정이 필요

Target 변수 살펴보기


```
data['Bankrupt?'].value_counts()
```

```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

파산을 안한 기업이 파산을 한 기업보다 훨씬 많다.

Target 변수 살펴보기

```
1 data['Bankrupt?'].value_counts()
```

```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

UNBALANCED DATASET

파산을 안한 기업이 파산을 한 기업보다 훨씬 많다.

결측치, 변수 자료형 확인

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                   Non-Null Count  Dtype
---  ------                                                   --------------  -----
 0   Bankrupt?                                                6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest 6819 non-null   float64
 2    ROA(A) before interest and % after tax                  6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax       6819 non-null   float64
 4    Operating Gross Margin                                  6819 non-null   float64
 5    Realized Sales Gross Margin                             6819 non-null   float64
 6    Operating Profit Rate                                   6819 non-null   float64
 7    Pre-tax net Interest Rate                               6819 non-null   float64
 8    After-tax net Interest Rate                             6819 non-null   float64
 9    Non-industry income and expenditure/revenue             6819 non-null   float64
 10   Continuous interest rate (after tax)                    6819 non-null   float64
 11   Operating Expense Rate                                  6819 non-null   float64
 12   Research and development expense rate                   6819 non-null   float64
 13   Cash flow rate                                          6819 non-null   float64
 14   Interest-bearing debt interest rate                     6819 non-null   float64
 15   Tax rate (A)                                            6819 non-null   float64
 16   Net Value Per Share (B)                                 6819 non-null   float64
 17   Net Value Per Share (A)                                 6819 non-null   float64
 18   Net Value Per Share (C)                                 6819 non-null   float64
 19   Persistent EPS in the Last Four Seasons                 6819 non-null   float64
 20   Cash Flow Per Share                                     6819 non-null   float64
 21   Revenue Per Share (Yuan ¥)                              6819 non-null   float64
 22   Operating Profit Per Share (Yuan ¥)                     6819 non-null   float64
```

결측치 없음

```
 84   Current Liability to Current Assets                     6819 non-null   float64
 85   Liability-Assets Flag                                   6819 non-null   int64
 86   Net Income to Total Assets                              6819 non-null   float64
 87   Total assets to GNP price                               6819 non-null   float64
 88   No-credit Interval                                      6819 non-null   float64
 89   Gross Profit to Sales                                   6819 non-null   float64
 90   Net Income to Stockholder's Equity                      6819 non-null   float64
 91   Liability to Equity                                     6819 non-null   float64
 92   Degree of Financial Leverage (DFL)                      6819 non-null   float64
 93   Interest Coverage Ratio (Interest expense to EBIT)      6819 non-null   float64
 94   Net Income Flag                                         6819 non-null   int64
 95   Equity to Liability                                     6819 non-null   float64
```

모두 수치형 변수

X85 - Liability-Assets Flag:
1 if Total Liability exceeds Total Assets, 0 otherwise
X94 - Net Income Flag:
1 if Net Income is Negative for the last two years, 0 otherwise

이미 one-hot encoding 되어 있음
=> 자료형 변환할 필요가 없다

이상치 처리

```python
def outliers_removal(feature,feature_name,dataset):

    # Identify 25th & 75th quartiles
    # Q1, Q3 값을 초과하거나 미만인 값들에 대해 이상치 제거 작업

    q25, q75 = np.percentile(feature, 25), np.percentile(feature, 75)
    #print('Quartile 25: {} | Quartile 75: {}'.format(q25, q75))
    feat_iqr = q75 - q25
    #print('iqr: {}'.format(feat_iqr))

    feat_cut_off = feat_iqr * 1.5
    feat_lower, feat_upper = q25 - feat_cut_off, q75 + feat_cut_off
    #print('Cut Off: {}'.format(feat_cut_off))
    #print(feature_name +' Lower: {}'.format(feat_lower))
    #print(feature_name +' Upper: {}'.format(feat_upper))

    outliers = [x for x in feature if x < feat_lower or x > feat_upper]
    #print(feature_name + ' outliers for close to bankruptcy cases: {}'.format(len(outliers)))
    #print(feature_name + ' outliers:{}'.format(outliers))

    dataset = dataset.drop(dataset[(dataset[feature_name] > feat_upper) | (dataset[feature_name] < feat_lower)].index)
    #print('-' * 65)

    return dataset

for col in data:
    df = outliers_removal(data[col],str(col),data)
```

Q1, Q3 값을 초과하거나 미만인 값들에 대해 이상치 제거

전

```python
data.shape
```

```
(6819, 96)
```

후

```python
df.shape
```

```
(6270, 96)
```

제거 후에도 데이터양이 충분

-> 대체가 아닌 제거

FEATURE SELECTION

Target 변수와 독립변수 간 상관관계 파악

```
1 df.corr()['Bankrupt?'].sort_values(ascending=False)
```

```
Bankrupt?                                                 1.000000
 Debt ratio %                                             0.260814
 Total debt/Total net worth                               0.229542
 Current Liability to Assets                              0.192470
 Borrowing dependency                                     0.173991
                                                            ...
 ROA(C) before interest and depreciation before interest -0.273792
 ROA(B) before interest and depreciation after tax       -0.286875
 ROA(A) before interest and % after tax                  -0.299326
 Net Income to Total Assets                              -0.330840
 Net Income Flag                                               NaN
Name: Bankrupt?, Length: 96, dtype: float64
```

눈에 띄게 큰 상관계수 값을 갖는 변수가 없어 종속변수에 크리티컬한 영향을 미치는 변수 판단 불가
=> 상관계수를 가지고 분석에 사용할 변수를 선택하기보다 다른 방법 생각

FEATURE SELECTION

TRAIN, TEST SPLIT

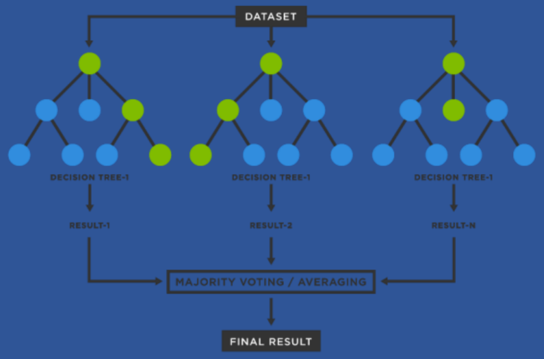
```
y = df['Bankrupt?']
x = df.drop(['Bankrupt?'], axis=1)
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

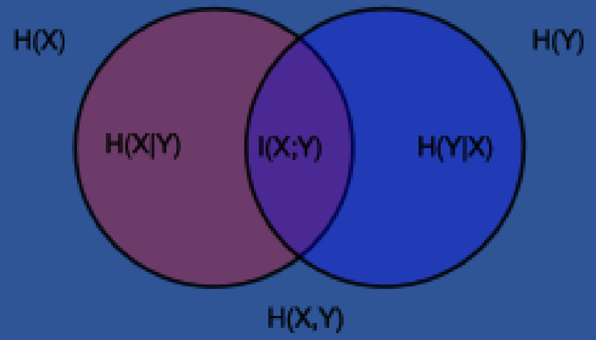
Data leakage를 방지하기 위해 본격적인 feature selection에 앞서 train, test split 진행

FEATURE SELECTION

Random Forest Feature Importance



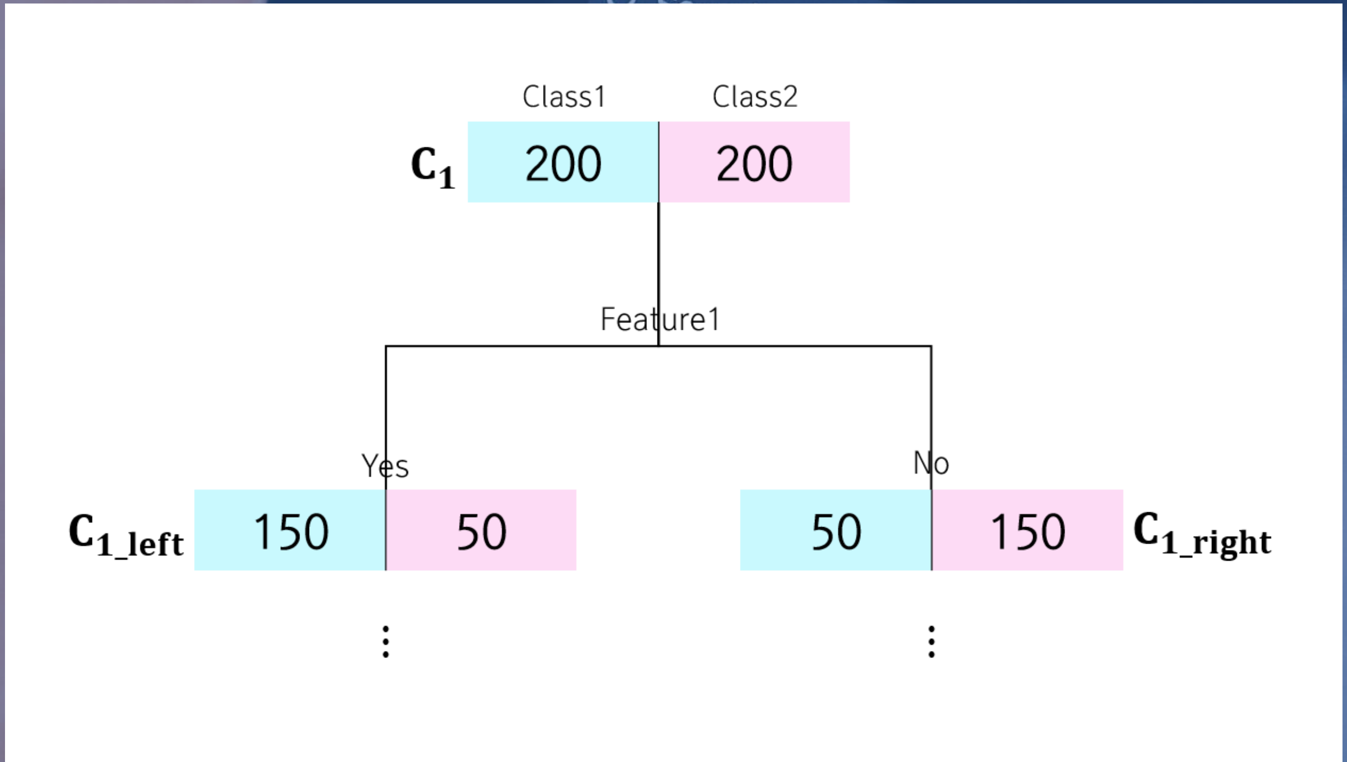
Mutual Information



PCA

FEATURE SELECTION

Random Forest Feature Importance



homogeneity가 증가한 정도인 **information Gain을 최대화**하는feature가 중요하다고 높다고 판단

FEATURE SELECTION

1. Feature Importance

```
model = RandomForestClassifier(n_estimators=1000, random_state=0, n_jobs=-1)
model.fit(x_train, y_train)
# 변수중요도 탐색
sel = SelectFromModel(model)
```

```
sel.fit(x_train, y_train)
```

```
SelectFromModel(estimator=RandomForestClassifier(n_estimators=1000, n_jobs=-1,
                                                 random_state=0))
```

```
selected_feat= x_train.columns[(sel.get_support())]
len(selected_feat)
```

```
38
```

```
selected_feat
```

```
Index([' ROA(C) before interest and depreciation before interest',
       ' ROA(A) before interest and % after tax',
       ' ROA(B) before interest and depreciation after tax',
       ' After-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Continuous interest rate (after tax)', ' Operating Expense Rate',
       ' Interest-bearing debt interest rate', ' Net Value Per Share (B)',
       ' Net Value Per Share (A)', ' Net Value Per Share (C)',
       ' Persistent EPS in the Last Four Seasons',
       ' Per Share Net profit before tax (Yuan ¥)', ' Net Value Growth Rate',
       ' Total Asset Return Growth Rate Ratio', ' Quick Ratio',
       ' Interest Expense Ratio', ' Total debt/Total net worth',
       ' Debt ratio %', ' Net worth/Assets', ' Borrowing dependency',
       ' Net profit before tax/Paid-in capital',
       ' Accounts Receivable Turnover', ' Average Collection Days',
       ' Fixed Assets Turnover Frequency', ' Working Capital to Total Assets',
       ' Cash/Total Assets', ' Cash/Current Liability',
       ' Inventory/Working Capital', ' Working Capital/Equity',
       ' Total income/Total expense', ' Net Income to Total Assets',
       ' No-credit Interval', ' Net Income to Stockholder's Equity',
       ' Liability to Equity', ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
       ' Equity to Liability'],
      dtype='object')
```

랜덤포레스트 결과

총 **38개**의 feature가 중요한 변수로 판단

FEATURE SELECTION

Mutual Information

$$\mathbb{I}(X;Y) \triangleq \mathbb{KL}\left(p(x,y) \| p(x)p(y)\right) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$

P(X, Y)가 p(X)p(Y)와 얼마나 비슷한지를 측정하는 척도로 X와 Y가 얼마나 서로 의존적인지 알 수 있음

FEATURE SELECTION

2. Mutual Information

```
mi = mutual_info_classif(x_train, y_train)
```

```
train_df = pd.DataFrame(x_train, columns = x_train.columns)
```

random forest feature importance를

사용한 선택한 변수 개수와 마찬가지로

상위 38개의 변수 추출

```
# select features
sel_ = SelectKBest(mutual_info_classif, k=38).fit(x_train, y_train)

# display features
train_df.columns[sel_.get_support()]
```

```
Index([' ROA(C) before interest and depreciation before interest',
       ' ROA(A) before interest and % after tax',
       ' ROA(B) before interest and depreciation after tax',
       ' Pre-tax net Interest Rate', ' After-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Continuous interest rate (after tax)', ' Tax rate (A)',
       ' Net Value Per Share (B)', ' Net Value Per Share (A)',
       ' Net Value Per Share (C)', ' Persistent EPS in the Last Four Seasons',
       ' Operating Profit Per Share (Yuan ¥)',
       ' Per Share Net profit before tax (Yuan ¥)', ' Current Ratio',
       ' Quick Ratio', ' Interest Expense Ratio',
       ' Total debt/Total net worth', ' Debt ratio %', ' Net worth/Assets',
       ' Borrowing dependency', ' Operating profit/Paid-in capital',
       ' Net profit before tax/Paid-in capital',
       ' Working Capital to Total Assets', ' Inventory/Working Capital',
       ' Working Capital/Equity', ' Current Liabilities/Equity',
       ' Retained Earnings to Total Assets', ' Total income/Total expense',
       ' Working capitcal Turnover Rate', ' Current Liability to Equity',
       ' Current Liability to Current Assets', ' Net Income to Total Assets',
       ' Net Income to Stockholder's Equity', ' Liability to Equity',
       ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
       ' Equity to Liability'],
      dtype='object')
```

FEATURE SELECTION

2. Mutual Information

```
columns = selected_feat|train_df.columns[sel_.get_support()]
len(columns)

```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:

Index.__or__ operating as a set operation is deprecated, in the future this will be a

48
```

```
x_train_last = x_train[columns]
```

```
x_test_last = x_test[columns]
```

위 두 가지 방법에 의해 추출된 feature들의 합집합(48개)을 변수로 하는 최종 데이터셋 생성

FEATURE SELECTION

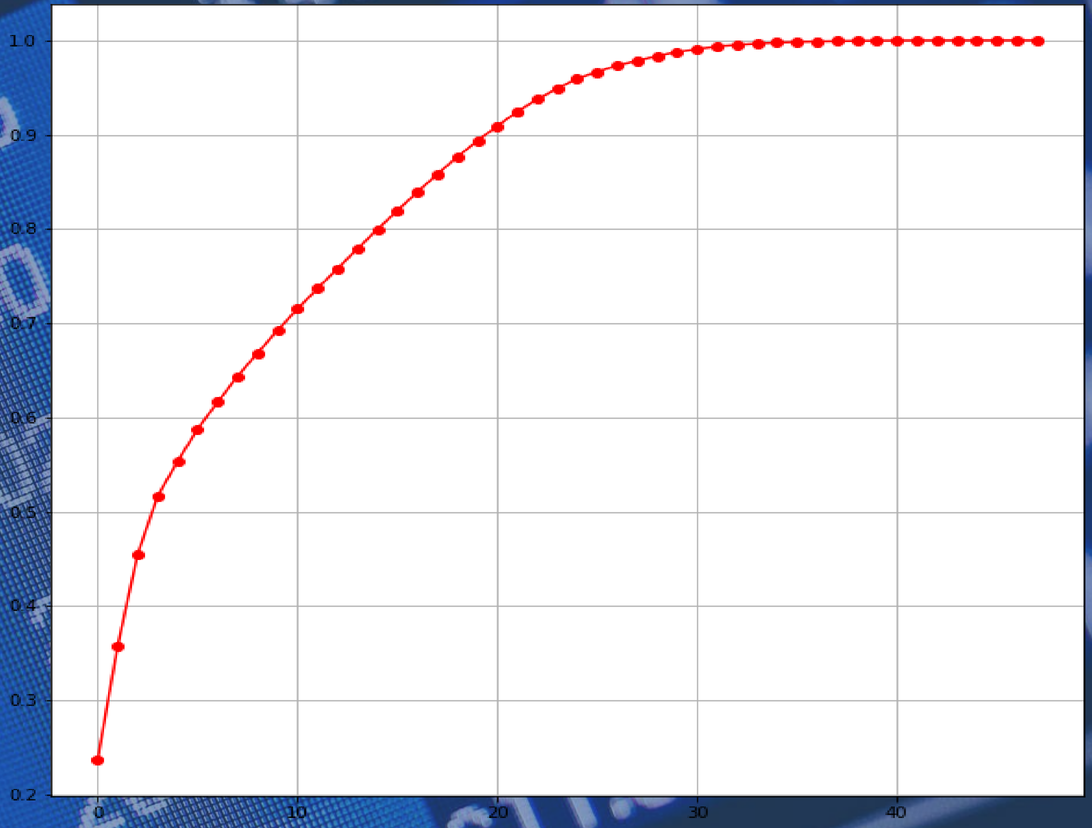
PCA 하기 전 표준화 진행

```python
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

x_train_scaled = ss.fit_transform(x_train_last)
x_test_scaled = ss.transform(x_test_last)
```

FEATURE SELECTION

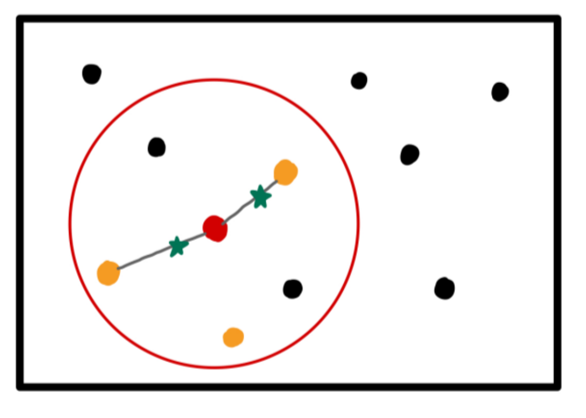
```python
from sklearn.decomposition import PCA

pca = PCA(random_state=0)
X_p = pca.fit_transform(x_train_scaled)

pd.Series(np.cumsum(pca.explained_variance_ratio_))
```

```
0     0.236847
1     0.357262
2     0.454617
3     0.516413
4     0.553366
5     0.587606
6     0.616220
7     0.643629
8     0.668355
9     0.692320
10    0.714978
11    0.736735
12    0.757979
13    0.778864
14    0.799416
15    0.819408
16    0.839085
17    0.858123
18    0.876443
19    0.893395
20    0.909027
```



분산의 80% 이상을 설명하는 변수 개수: **15개**

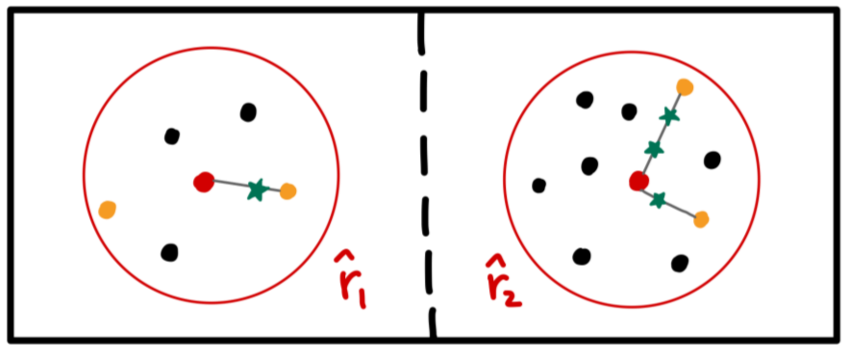
UNBALANCED DATASET



SMOTE

k-nearest neighbors(knn) 이용

먼저 knn으로 가까운 minority class 들을 찾은 후 0과 1사이의 랜덤한 값으로 내분하는 점을 새로운 샘플로 생성하는 기법

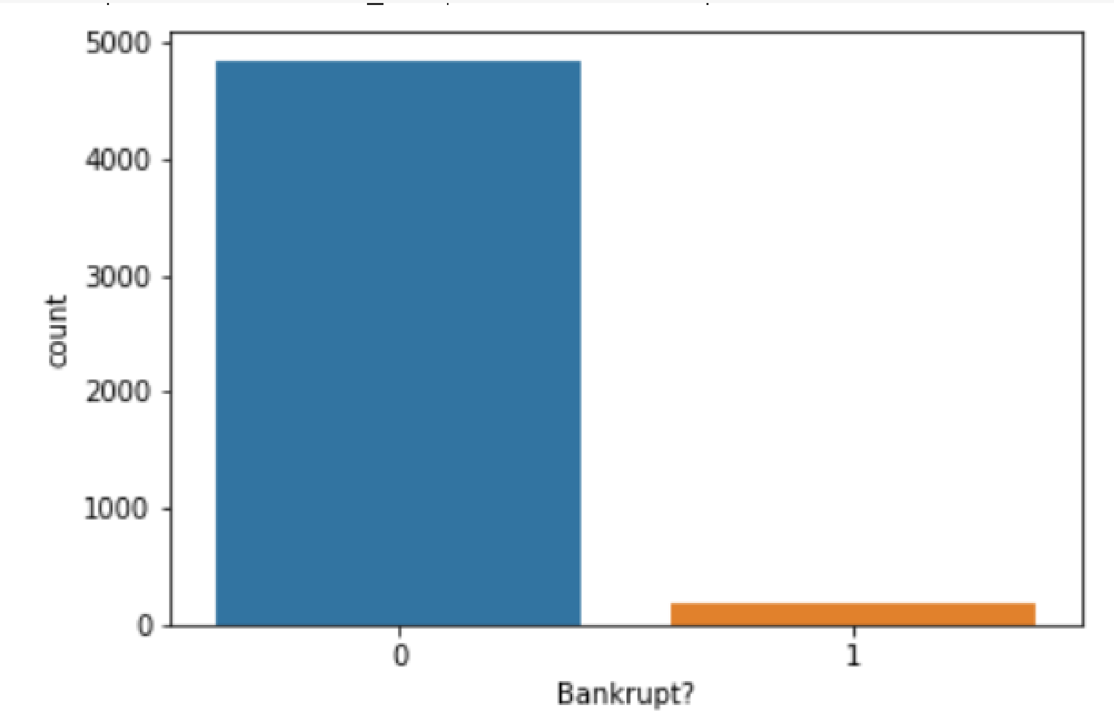


ADASYN

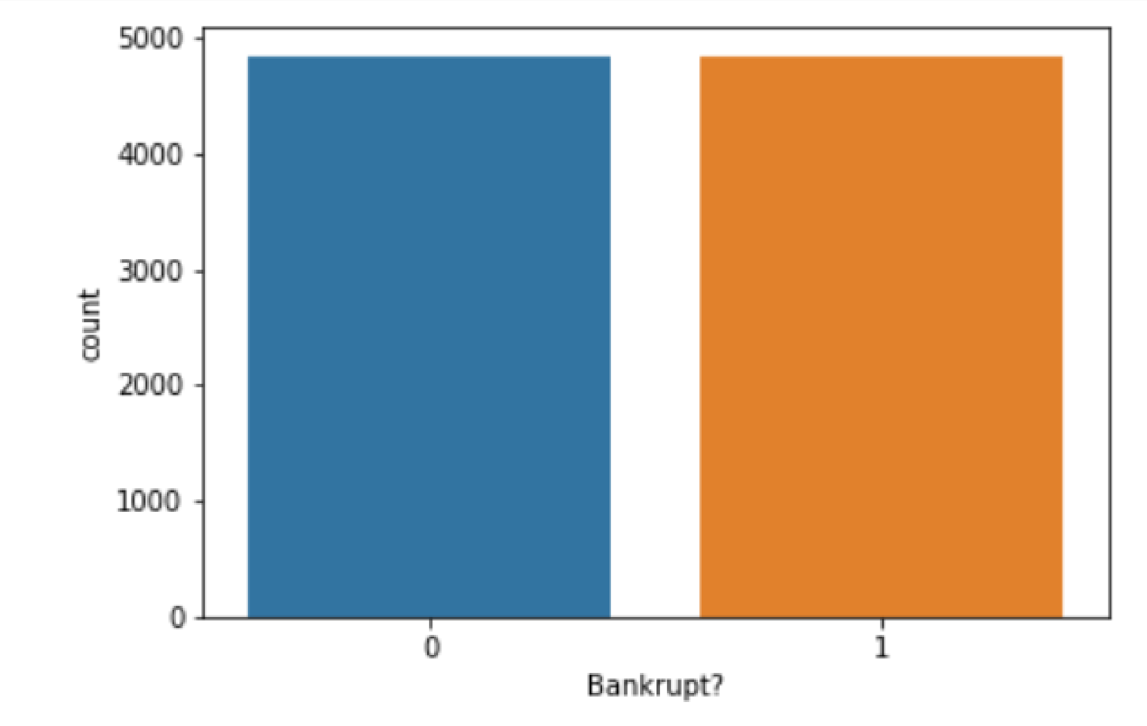
SMOTE와 달리 각 관측치마다 weight를 부여하여, 생성하는 샘플의 수가 다르다

이 weight는 knn 범위 내로 들어오는 majority class의 개수에 비례

UNBALANCED DATASET



전



후

MODELING

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine
- Naïve Bayes
- XGBoost

\+

Grid Search

Ensemble(voting)

MODELING

결과

Random forest를 이용해 0.9741 정확도 산출

```
# RandomForest GridSearch
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

param_grid = [
              {'n_estimators':[3,10,30,35,40], 'max_features':[2,4,6,8,10,12]},
              {'bootstrap':[False],'n_estimators':[3,5,10], 'max_features':[2,3,4]}
]

forest_reg = RandomForestClassifier()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='accuracy',
                           verbose=1,
                           return_train_score=True)


# grid_search.fit(X_train_pca, y_train)
grid_search.fit(X_train_over, y_train_over) # smote
```

```
Fitting 5 folds for each of 39 candidates, totalling 195 fits
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid=[{'max_features': [2, 4, 6, 8, 10, 12],
                          'n_estimators': [3, 10, 30, 35, 40]},
                         {'bootstrap': [False], 'max_features': [2, 3, 4],
                          'n_estimators': [3, 5, 10]}],
             return_train_score=True, scoring='accuracy', verbose=1)
```

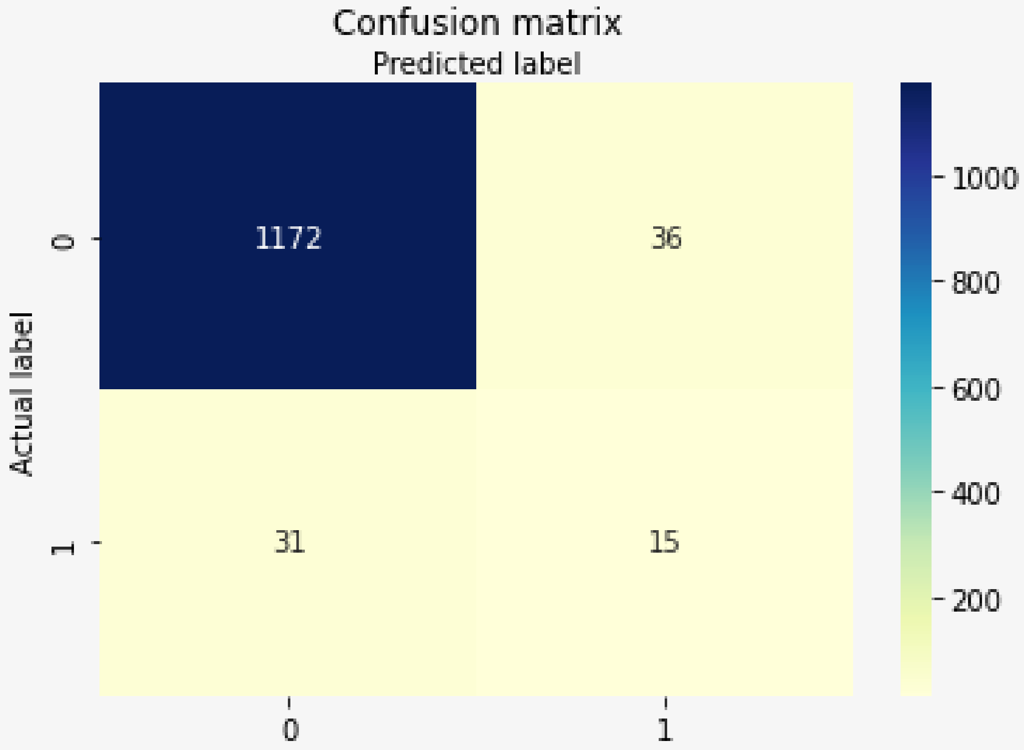
```
print(grid_search.best_params_)
final_model = grid_search.best_estimator_
```

```
{'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
```

```
print('최적 하이퍼 파라미터:\n', grid_search.best_params_)
print('최고 예측 정확도: {0:.4f}'.format(grid_search.best_score_))
```

```
최적 하이퍼 파라미터:
 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
최고 예측 정확도: 0.9741
```

소감 및 발전 방향



Unbalanced dataset 문제를 해결할 수 있는 다양한 기법을 다루어볼 수 있었음. 다만 oversampling을 적용했음에도 True Negative가 적게 나왔음

변수 선택과정에 있어서
도메인 지식의 중요성을 깨달음