ESC FINAL PROJECT

Bankruptcy Classification – EDA까지

TABLE



조원 소개



데이터 및 프로젝트 소개



EDA

- NA Imputation
- Factor Analysis



추후 계획





조원 소개

ESC 5조



김윤전

5조 조장!



김내히

5조의 유일한 신입!



정재은

5조의 에이스, 파잘알!



홍익선

왠지 부조장의 느낌!



엄상준

늙은이





데이터 개수(Observations): 9792

변수 개수: 65

Target Class: 파산(1) / 정상(0)

파산기업 수: 515, 정상기업 수: 9277

변수

X1 net profit / total assets

X2 total liabilities / total assets 등과 같이 재무상태표의 내용을 조합한 수치들.

프로젝트 목표

- 변수들을 통해 기업의 파산여부 예측
 - 기업의 파산여부에 어떠한 변수들이 중요한지 확인



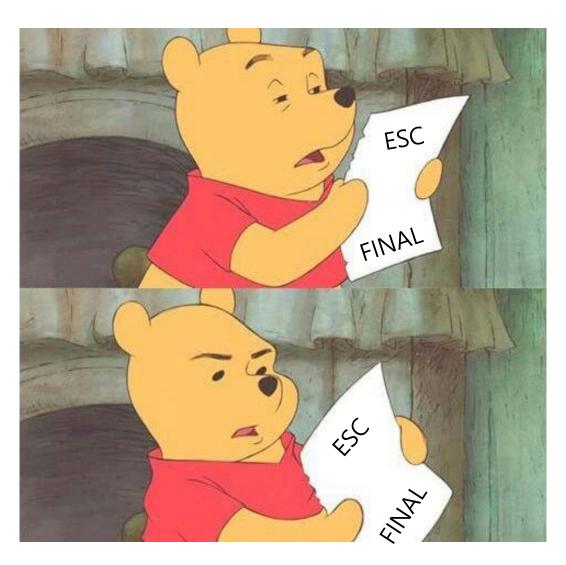


Classification: 재무상태표와 2년 후 파산 여부

- 9792행, 65열, Taget 'class' (파산 기업 1, 정상 기업 0)
- Train-Test split 7 대 3 (stratified split, train만 제공)
- 결측치 없음
- 당신은 지중은행 여신담당부서에 데이터 분석 직군으로 입사한 신입사원입니다. 당신의 역할은 기업의 재무상태표를 보고 파산 확률을 예측해 여신 여부를 결정하는 것입니다. 당신의 상사가 당신에게 기대하는 바는 다음과 같습니다.
 - i. 기업의 재무상태표, 손익계산서, 현금흐름표 등 각종 항목이 서로 어떤 관련이 있는가?
 - ii. 기업의 2년 후 파산 여부를 재무상태표 항목으로 보고 예측할 수 있는가?
 - iii. 기업의 재무상태표에서 여신 여부를 결정하기 위해 무엇을 중점으로 봐야 하는가?

20-1 ESC FINAL PROJECT 설명에서 발췌





print(f"Shape: {df.shape}")
print(df.isnull().sum())

Attr50

Attr51

Attr52

Attr53 Attr54

Attr55

Attr56

Attr57

Attr58

Attr59

Attr60

Attr61

Attr62

Attr63

Attr64

dtype: int64

class

15

0

60

162

162

0

14

10

1

420

20

14

28

162

0

Shape:	(6855, 65)	Attr2	5 0
Attr1	0	Attr2	
Attr2	0	Attr2	
Attr3	0	Attr2	8 162
Attr4	28	Attr2	9 0
Attr5	15	Attr3	0 14
Attr6	0	Attr3	1 14
Attr7	0	Attr3	2 72
Attr8	15	Attr3	3 28
Attr9	0	Attr3	4 15
Attr10	0	Attr3	5 0
Attr11	0	Attr3	6 0
Attr12	28	Attr3	7 3100
Attr13	14	Attr3	8 0
Attr14	0	Attr3	9 14
Attr15	5	Attr4	0 28
Attr16	15	Attr4	1 142
Attr17	15	Attr4	2 14
Attr18	0	Attr4	3 14
Attr19	14	Attr4	4 14
Attr20	14	Attr4	5 418
Attr21	112	Attr4	6 28
Attr22	0	Attr4	7 57
Attr23	14	Attr4	8 0
Attr24	149	Attr4	9 14



Recovering Some Variables

- 대부분의 column에서 결측치가 발견되었다
- NA imputation 을 위해, 최대한 구할 수 있는 변수 값들은 구해보자. (NA 갯수가 0인 Attr 들을 이용)

EX)

```
total_assets = pd.Series(np.exp(df['Attr29']), index=df['Attr29'].index)
assert not bool(total_assets.isnull().sum())

total_equity = df['Attr10']* total_assets
assert not bool(total_equity.isnull().sum())

total_liabilities = total_assets - total_equity
assert not bool(total_liabilities.isnull().sum())

short_term_liabilities = df['Attr51'] * total_assets
assert not bool(short_term_liabilities.isnull().sum())

long_term_liabilities = total_liabilities.isnull().sum())

total_sales = pd.Series(np.array(short_term_liabilities) * np.array(df['Attr36']))
assert not bool(total_sales.isnull().sum())
```

구할 수 있었던 정보들

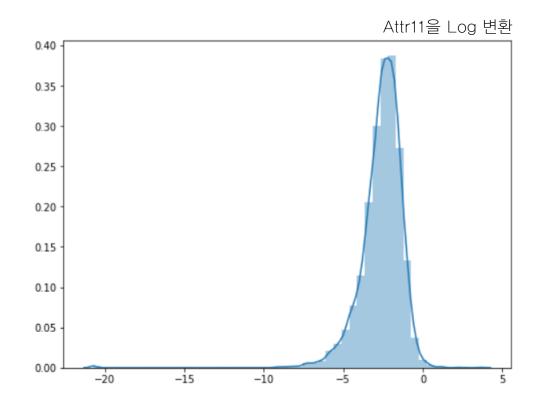
```
total_assets
total_equity
total_liabilities
short_term_liabilities
long_term_liabilities
total_sales
Sales
gross_propit
retained_earnings
EBIT
EBITDA
net_profit
working_capital
profit_on_sales
```





Delete Some Outliers

- 더 좋은 Imputation을 위해 Outlier도 제거해주자.



df['Attr11'].sort_values(ascending=True).index[0]

423 423 번째 데이터가 문제군! → 빼자

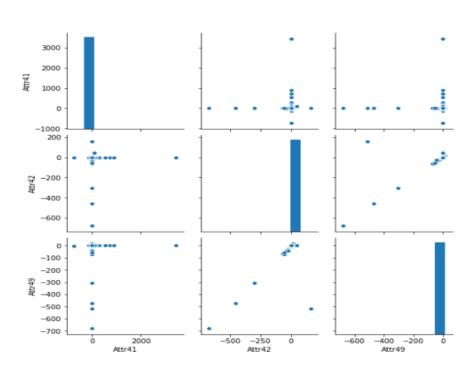
이런 식으로 density plot을 본 이후 수작업으로
 아웃라이어들이 포함된 관측치를 삭제해준다.

 하지만 class 1에 해당하는 관측치가 매우 적으니 아웃라이어여도 class 1,
 즉 부도난 회사의 데이터면 삭제하지 않는다.



Drop Useless Columns

- 변수가 가지고 있는 정보가 겹치거나, 눈으로 보았을 때 명확한 선형관계가 존재하면 제거해준다.



Scatter Plot 확인

버린 변수들

- X15 (total liabilities * 365) / (gross profit + depreciation)
- X17 total assets / total liabilities
- X19 gross profit / sales
- X20 (inventory * 365) / sales
- X28 working capital / fixed assets
- X31 (gross profit + interest) / sales
- X34 operating expenses / total liabilities
- X37 (current assets inventories) / long-term liabilities
- X41 total liabilities / ((profit on operating activities + depreciation) * (12/365))
- X42 profit on operating activities / sales
- X44 (receivables * 365) / sales
- X46 (current assets inventory) / short-term liabilities
- X55 working capital
- X57 (current assets inventory short-term liabilities) / (sales gross profit depreciation)
- X62 (short-term liabilities *365) / sales



Direct NA Imputation using recovered information

- 분모가 0이라서 NA값으로 뜨는 값들이 있었다.
- 이런 애들 중 아까 직접 구했던 정보들로 채울 수 있는 애들은 채워주자.

● 분모가 0이 되지 않기 위해 0 대신 최솟값 대입

```
min_short_term_liabilities = [x for x in np.sort(short_term_liabilities) if x][0]
short_term_liabilities.replace(0, min_short_term_liabilities, inplace=True)

min_sales = [x for x in np.sort(sales) if x][0]
sales.replace(0, min_sales, inplace=True)

min_total_liabilities = [x for x in np.sort(total_liabilities) if x][0]
total_liabilities.replace(0, min_total_liabilities, inplace=True)

min_total_equity = [x for x in np.sort(total_equity) if x][0]
total_equity.replace(0, min_total_equity, inplace=True)
```

● 최솟값 대입 후 가진 정보들로 다시 계산

```
df['Attr12'] = gross_profit / short_term_liabilities
df['Attr23'] = net_profit / sales
df['Attr39'] = profit_on_sales / sales
df['Attr49'] = EBITDA / sales
df['Attr50'] = current_assets / total_liabilities
df['Attr59'] = long_term_liabilities / total_equity
df['Attr63'] = sales / short_term_liabilities
```

● NA값 확인 → 없다!



Attr12 0
Attr23 0
Attr39 0
Attr49 0
Attr50 0
Attr59 0
Attr63 0
dtype: int64



Using packages specialized in NA Imputation



그냥 NA 있는 Data들을 다 빼고 하면 되지 않아요?

Measuring performance of Logistic Regression with na removed data

```
na_dropped = df.dropna(axis = 0)
X, y = na_dropped.iloc[:, :-1], na_dropped.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, stratify = y, random_state=5)

Ir = LogisticRegression(class_weight='balanced')
Ir.fit(X_train, y_train)

y_pred = Ir.predict(X_test)
print(f1_score(y_test, y_pred))
```

0.08823529411764706

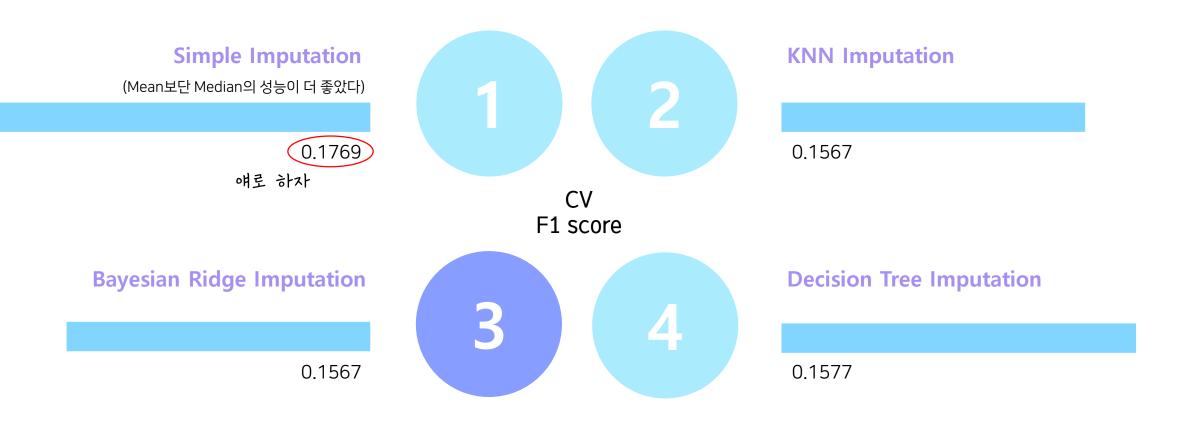
매우 낮다.



우주야 파이널이 생각보다 녹록지 않다..



Using packages specialized in NA Imputation





Using packages specialized in NA Imputation

Median Imputer NA 확인

Attr2 0 Attr35 0 Attr3 Attr36 0 Attr38 0 Attr4 Attr39 0 Attr5 Attr6 Attr40 0 Attr7 Attr43 0 0 Attr8 Attr45 Attr9 Attr47 0 Attr10 Attr48 0 Attr11 Attr49 0 Attr12 Attr50 0 Attr13 Attr51 0 Attr14 Attr52 0 Attr16 Attr53 0 Attr18 Attr54 0 Attr21 Attr56 0 Attr22 Attr58 0 Attr23 Attr59 0 Attr24 Attr60 0 Attr25 Attr61 0 Attr26 Attr63 0 Attr27 Attr64 0 Attr29 class 0

Attr1

Attr30

Attr32

0

깔끔.

Attr33

dtype: int64





그래도 여전히 변수가 너무 많다.(49개) 그런데 변수 꼴을 보아하니 변수들끼리 상관 관계가 있을 것 같다..



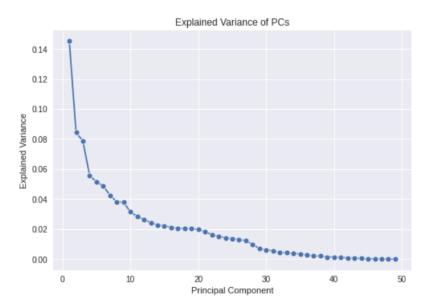


Scaling for Factor Analysis

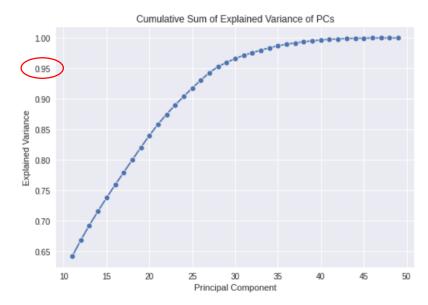
- Factor Analysis를 하기 위해서는 Scaling이 우선.
- Scikitlearn에서 StandardScaler 사용

Draw Scree Plot using PCA

- FA에서 n_factor를 위해 우선 PCA Scree Plot을 그려보자



Explained Variance of PCs



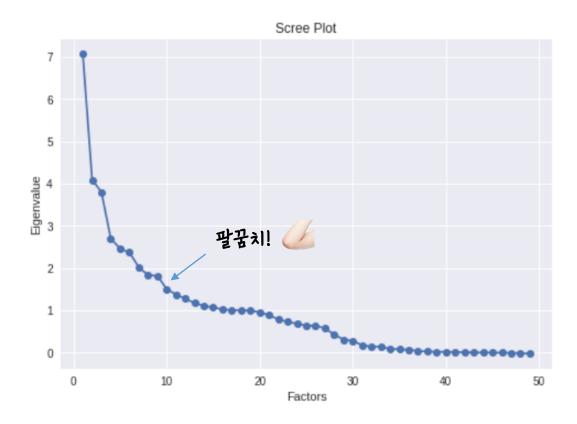
Cumulative Sum of Explained Variance of PCs

한.. 30개 정도면 충분할 것 같다.



Draw Scree Plot using FA

- 이번에는 FactorAnalyzer를 이용해서 Scree plot을 그리자.



10개 정도의 Factor로 데이터를 설명할 수 있을 것 같다.



Factor Analysis

- Factor Analysis 를 통해 Factor에 어떤 변수들이 묶였는지 확인해보자.
- 우선 용이한 시각화를 위해 Factor 두 개만 살펴보자.

```
for i in range(1, 3):
    tmp = fa_df.iloc[i, :]
    top3 = list(np.argsort(-tmp.values)[:3].reshape(3,))
    bottom3 = list(np.argsort(tmp.values)[:3].reshape(3))
    print("Factor"+str(i))
    print(f"Top3 is {real_columns[top3[0]]}, {real_columns[top3[1]]}, {real_columns[top3[2]]}")
    print(f"Bottom3 is {real_columns[bottom3[0]]}, {real_columns[bottom3[1]]}, {real_columns[bottom3[2]]}")
    if i == 1:
        print("=" * 50)
```

Factor1

Top3 is equity / total assets, (equity - share capital) / total assets, constant capital / total assets

Bottom3 is total liabilities / total assets, short-term liabilities / total assets, (total liabilities - cash) /
sales

Factor2

Top3 is (gross profit + depreciation) / total liabilities, (net profit + depreciation) / total liabilities, oper ating expenses / short-term liabilities

Bottom3 is logarithm of total assets, short-term liabilities / total assets, profit on sales / total assets

잘 모르겠지만..

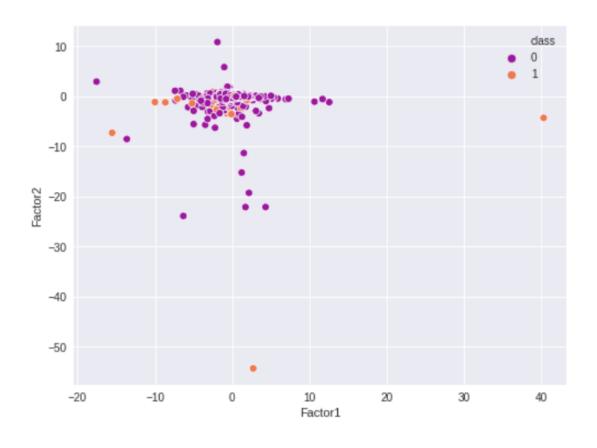
Factor1, Factor2 모두

재정건전성과 관련이 있는 것 같다..



Factor Analysis

- Factor Analysis 를 통해 Factor에 어떤 변수들이 묶였는지 확인해보자.



- 부도가 난 회사들은 대부분 Factor1의 부분과, Factor2의 0 부분에 위치하는 것으로 보아 재정건전성 이 부족해 보인다
- 물론 Outlier도 존재.



- Factor Analysis
 - Factor Analysis 를 통해 Factor에 어떤 변수들이 묶였는지 확인해보자.
- 마지막으로 어떤 변수들의 정보량이 Factor에 많이 반영되었는지 알아보자.

```
top10 = np.argsort(-fa.get_communalities())[:10]

for t in top10:
    print(real_columns[t])

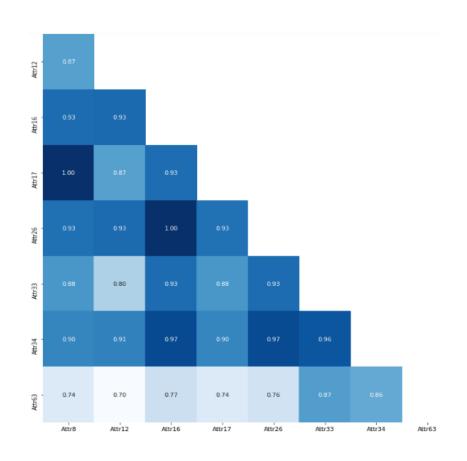
sales (n) / sales (n-1)
profit on operating activities / financial expenses
long-term liabilities / equity
[(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] *
365
sales / receivables
net profit / sales
total costs /total sales
EBITDA (profit on operating activities - depreciation) / sales
(sales - cost of products sold) / sales
sales / total assets
```

→ Liablities, Profit, Sales 관련된 변수들이 많이 반영되었음을 알 수 있다.





변수 추가 제거



Corr Matrix를 깜박하고 나중에 확인해봤더니 Correlation이 매우 높은 애들이 있더라!

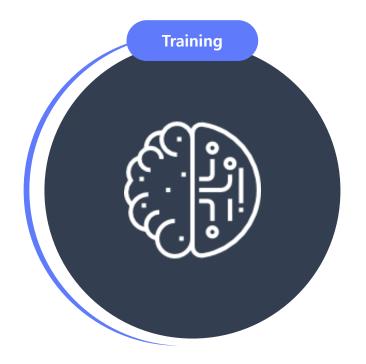


- --> FA 이전에 변수를 추가적으로 제거 해주려 함.
- --> 그 다음에는 우리가 했던 방식 그대로 다시 진행.



추후 계획

Future Plan



사용 가능한 모든 모델을 동원해서 모델을 학습 Ensemble 등..



모델을 선택한다. - 모델 자체의 선택

- 모델 내 변수 선택



모델의 예측성능 평가와 모델 해석

