

KAIST MFE, 2025 Spring

Kim Hyeonghwan

2025-03-05

Table of contents

Welcome!	4
I 머신러닝2('25 봄)	5
1 빅데이터와 금융자료 분석 CH1	6
2 빅데이터와 금융자료 분석 기말대체과제	17
2.1 Question 1	17
2.1.1 (1)	17
2.1.2 (2)	21
2.1.3 (3)	25
2.2 Question 2	28
2.2.1 (1)	28
2.2.2 (2)	31
2.2.3 (3)	32
2.2.4 (4)	33
2.2.5 (5)	33
2.3 Question 3	35
2.3.1 (1)	35
2.3.2 (2)	37
빅데이터와 금융자료분석 프로젝트 (Team 4)	39
1. 프로젝트 개요	39
2. 데이터의 구조, 특성 (EDA)	39
데이터의 수집, 기본구조	39
데이터의 특성	40
3. 데이터의 전처리	42
4. 대출 연체여부 예측 모델 구축 및 평가	43
ADASYN을 이용한 오버샘플링	43
XGBoost 모델 구축	43
모델 일반화성능 평가	43
변수 중요도(Feature Importance) 분석	45
시사점	46

II 알고리즘 거래전략('25 봄)	47
알고리즘 거래전략과 고빈도 금융	48
III 사례로 보는 금융공학('25 봄)	49
사례로 보는 금융공학 실무	50
코스피200 변동성 조정 위클리 양매도 전략	51
1. 개요	51
2. 배경지식	51
양매도 (Short strangle)	51
코스피200 변동성지수 (V-Kospi200 index)	52
코스피200 위클리옵션	52
3. 전략 소개 및 구현	53
전략 개요	53
행사가격 범위 설정방법	53
포트폴리오 구성 방법	54
3. Backtesting	54
데이터 수집 및 전처리, 포트폴리오 구현	54
Backtesting 성과	57
한계점	59
4. 포트폴리오 검증 ('25.3.13 ~ 4.10)	59
5. Appendix : Python, R code	61
Pyhon code : 데이터 수집 및 전처리	61
R code 1 : 데이터 가공, Backtesting	62
R code 2 : 전략 검증	71
IV 장외파생상품 기초 실무('25 봄)	77
장외파생상품 기초 실무	78
V 금융윤리와 사회책임('25 봄)	79
금융윤리와 사회책임	80

Welcome!

안녕하세요, KAIST MFE 25년 봄학기에 이수한 과목의 과제 등을 정리해두었습니다.

Part I

머신러닝2('25 봄)

1 빅데이터와 금융자료 분석 CH1

```
import numpy as np
import pandas as pd

missdict = {'f1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
            'f2': [10., None, 20., 30., None, 50., 60., 70., 80., 90.],
            'f3': ['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'C', 'C']}
missdata = pd.DataFrame( missdict )
missdata.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  -- 
 0   f1      10 non-null    int64  
 1   f2      8 non-null    float64 
 2   f3      10 non-null    object  
dtypes: float64(1), int64(1), object(1)
memory usage: 368.0+ bytes

missdata.isna().mean()

f1    0.0
f2    0.2
f3    0.0
dtype: float64

tmpdata1 = missdata.dropna()
tmpdata1
```

	f1	f2	f3
0	1	10.0	A
2	3	20.0	A
3	4	30.0	A
5	6	50.0	B
6	7	60.0	B
7	8	70.0	B
8	9	80.0	C
9	10	90.0	C

```
tmpdata2 = missdata.dropna( subset=['f3'] )
tmpdata2
```

	f1	f2	f3
0	1	10.0	A
1	2	NaN	A
2	3	20.0	A
3	4	30.0	A
4	5	NaN	B
5	6	50.0	B
6	7	60.0	B
7	8	70.0	B
8	9	80.0	C
9	10	90.0	C

```
numdata = missdata.select_dtypes(include=['int64', 'float64'])
tmpdata3 = numdata.fillna( -999, inplace=False )
tmpdata3.describe()
```

	f1	f2
count	10.000000	10.000000
mean	5.500000	-158.800000
std	3.02765	443.562297
min	1.000000	-999.000000
25%	3.25000	12.500000
50%	5.50000	40.000000
75%	7.75000	67.500000

	f1	f2
max	10.00000	90.000000

```
numdata.mean()
```

```
f1      5.50
f2     51.25
dtype: float64
```

```
tmpdata4 = numdata.fillna( numdata.mean(), inplace=False )
tmpdata4
```

	f1	f2
0	1	10.00
1	2	51.25
2	3	20.00
3	4	30.00
4	5	51.25
5	6	50.00
6	7	60.00
7	8	70.00
8	9	80.00
9	10	90.00

```
missdata.groupby('f3')['f2'].mean()
```

```
f3
A    20.0
B    60.0
C    85.0
Name: f2, dtype: float64
```

```
missdata.groupby('f3')['f2'].transform('mean')
```

```
0    20.0
1    20.0
2    20.0
```

```
3    20.0
4    60.0
5    60.0
6    60.0
7    60.0
8    85.0
9    85.0
Name: f2, dtype: float64
```

```
tmpdata5 = numdata.copy()
tmpdata5['f2'].fillna( missdata.groupby('f3')['f2'].transform('mean'), inplace=True)
tmpdata5
```

```
/var/folders/n2/jbh_0_091bx8qgz7j87t2qwc0000gp/T/ipykernel_25894/622840210.py:2: FutureWarning: A
The behavior will change in pandas 3.0. This inplace method will never work because the intermedia
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value},

```
tmpdata5['f2'].fillna( missdata.groupby('f3')['f2'].transform('mean'), inplace=True)
```

	f1	f2
0	1	10.0
1	2	20.0
2	3	20.0
3	4	30.0
4	5	60.0
5	6	50.0
6	7	60.0
7	8	70.0
8	9	80.0
9	10	90.0

```
missdata_tr = missdata.dropna()
x_tr = missdata_tr[['f1']]
y_tr = missdata_tr['f2']

from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

```

model.fit( x_tr, y_tr )

missdata_ts = missdata [ missdata.isnull().any(axis=1) ]
x_ts = missdata_ts[['f1']]

predicted_values = model.predict( x_ts )
tmpdata6 = missdata.copy()
tmpdata6.loc[ tmpdata6['f2'].isnull(), 'f2' ] = predicted_values
tmpdata6

```

	f1	f2	f3
0	1	10.000000	A
1	2	14.191176	A
2	3	20.000000	A
3	4	30.000000	A
4	5	41.985294	B
5	6	50.000000	B
6	7	60.000000	B
7	8	70.000000	B
8	9	80.000000	C
9	10	90.000000	C

```

missdata_num = missdata.copy()
missdata_num['f3']=missdata_num['f3'].map({'A':1,'B':2,'C':3})

missdata_num

```

	f1	f2	f3
0	1	10.0	1
1	2	NaN	1
2	3	20.0	1
3	4	30.0	1
4	5	NaN	2
5	6	50.0	2
6	7	60.0	2
7	8	70.0	2
8	9	80.0	3
9	10	90.0	3

```

from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
tmpdata7 = imputer.fit_transform(missdata_num)

pd.DataFrame( tmpdata7 )

```

	0	1	2
0	1.0	10.0	1.0
1	2.0	15.0	1.0
2	3.0	20.0	1.0
3	4.0	30.0	1.0
4	5.0	40.0	2.0
5	6.0	50.0	2.0
6	7.0	60.0	2.0
7	8.0	70.0	2.0
8	9.0	80.0	3.0
9	10.0	90.0	3.0

```

outdict = {'A': [10, 0.02, 0.3, 40, 50, 60, 712, 80, 90, 1003],
           'B': [0.05, 0.00015, 25, 35, 45, 205, 65, 75, 85, 3905]}
outdata = pd.DataFrame( outdict )

Q1 = outdata.quantile(0.25)
Q3 = outdata.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

((outdata < lower_bound) | (outdata > upper_bound))

```

	A	B
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	True
6	True	False

	A	B
7	False	False
8	False	False
9	True	True

```
outliers = ((outdata < lower_bound) | (outdata > upper_bound)).any(axis=1)
outliersdata = outdata[ outliers ]
outliersdata
```

	A	B
5	60.0	205.0
6	712.0	65.0
9	1003.0	3905.0

```
standardizeddata = (outdata - outdata.mean()) / outdata.std()
standardizeddata
```

	A	B
0	-0.552206	-0.364647
1	-0.580536	-0.364688
2	-0.579741	-0.344154
3	-0.467047	-0.335940
4	-0.438661	-0.327727
5	-0.410274	-0.196309
6	1.440519	-0.311300
7	-0.353501	-0.303086
8	-0.325115	-0.294872
9	2.266563	2.842723

```
outliers2 = ((standardizeddata < -3) | (standardizeddata > 3)).any(axis=1)
outliersdata2 = outdata[ outliers2 ]
outliersdata2
```

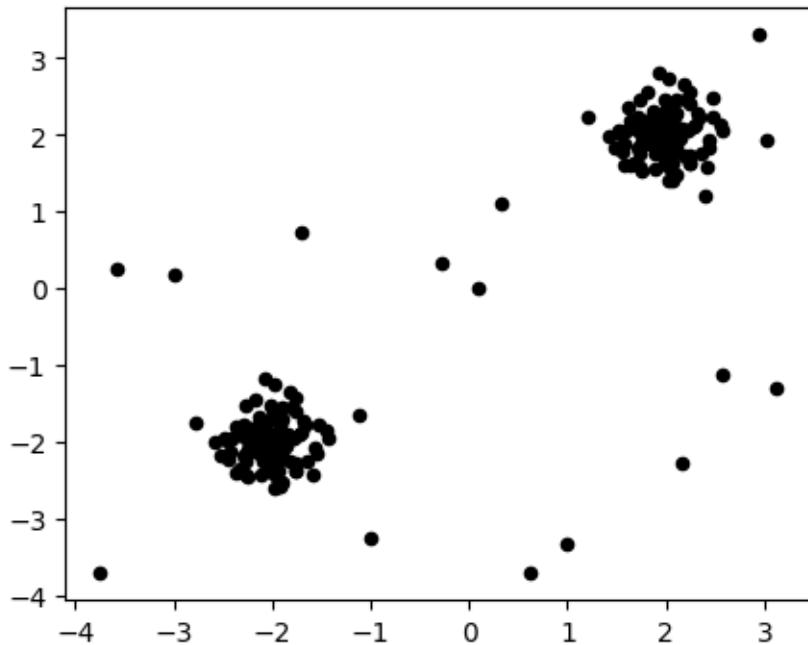
	A	B

```

import matplotlib.pyplot as plt
np.random.seed(42)
X_inliers = 0.3 * np.random.randn(100, 2)
X_outliers = np.random.uniform(low=-4, high=4, size=(20, 2))
X = np.r_[X_inliers + 2, X_inliers - 2, X_outliers]

plt.figure(figsize=(5, 4))
plt.scatter(X[:, 0], X[:, 1], color='k', s=20)

```

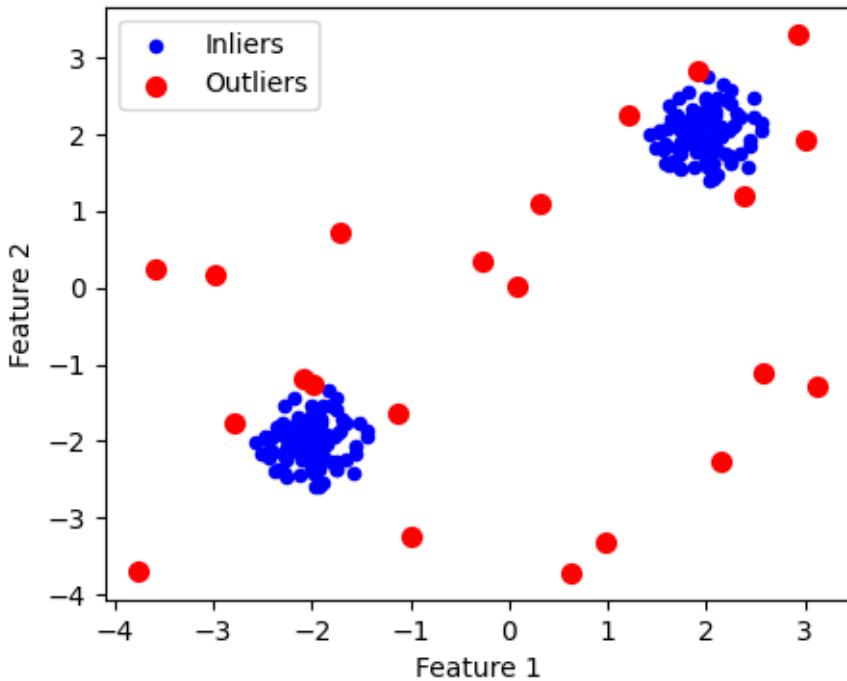


```

from sklearn.neighbors import LocalOutlierFactor
clf = LocalOutlierFactor(n_neighbors=20, contamination=0.1)
y_pred = clf.fit_predict(X) # 1: inlier, -1: outlier
outlier_mask = y_pred == -1

plt.figure(figsize=(5, 4))
plt.scatter(X[:, 0], X[:, 1], color='b', s=20, label='Inliers')
plt.scatter(X[outlier_mask, 0], X[outlier_mask, 1], color='r', s=50, label='Outliers')
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()

```



```

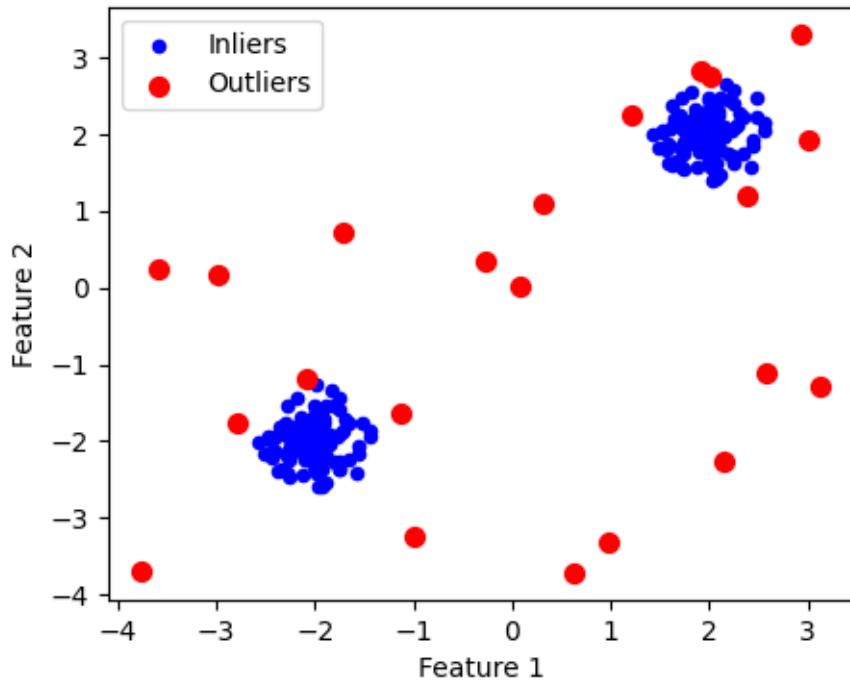
from sklearn.ensemble import IsolationForest
clf2 = IsolationForest(contamination=0.1)

# contamination : 이상치 비율
# n_estimators : 나무의 갯수 (defalut 100)
# max_features : 각 나무별 특성변수의 갯수(default 1)
clf2.fit( X )

y_pred2 = clf2.predict( X ) # 1: inlier, -1: outlier
outlier_mask2 = y_pred2 == -1

plt.figure(figsize=(5, 4))
plt.scatter(X[:, 0], X[:, 1], color='b', s=20, label='Inliers')
plt.scatter(X[outlier_mask2, 0], X[outlier_mask2, 1], color='r', s=50,label='Outliers')
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()

```



```
clf2.score_samples(X)
```

```
array([-0.39567479, -0.43973362, -0.38576701, -0.4625295 , -0.39169087,
       -0.39353101, -0.44632054, -0.4539587 , -0.39410011, -0.42823486,
       -0.43802322, -0.4231585 , -0.38600184, -0.40324911, -0.38778406,
       -0.46849881, -0.4075734 , -0.43420431, -0.45140279, -0.4113938 ,
       -0.40036102, -0.38335266, -0.42666089, -0.41104579, -0.44476313,
       -0.38744281, -0.39942914, -0.44257912, -0.38938554, -0.40470075,
       -0.39006555, -0.42881353, -0.44275864, -0.40154039, -0.39729665,
       -0.42434279, -0.42743206, -0.57699503, -0.38628797, -0.45454533,
       -0.3864026 , -0.44513991, -0.39598029, -0.41231495, -0.39270763,
       -0.40568073, -0.39005843, -0.43210962, -0.38601781, -0.38485125,
       -0.41991455, -0.40181793, -0.38788591, -0.48400217, -0.38538727,
       -0.47693547, -0.50815903, -0.39115267, -0.40423505, -0.43216634,
       -0.4254748 , -0.48475901, -0.4890737 , -0.40357175, -0.39439224,
       -0.42406868, -0.40171635, -0.44870204, -0.38761922, -0.43170548,
       -0.42364173, -0.43593615, -0.39560581, -0.4391784 , -0.39543452,
       -0.38550106, -0.38910014, -0.39558501, -0.48693408, -0.41865632,
       -0.40964165, -0.43730378, -0.41716448, -0.47893783, -0.39791987,
       -0.40492088, -0.38431516, -0.39544321, -0.42208103, -0.53522817,
       -0.41839783, -0.40171635, -0.39362168, -0.39333773, -0.44128807,
       -0.40208128, -0.40907841, -0.39096216, -0.38929625, -0.40645206,
       -0.38953796, -0.43147334, -0.38691831, -0.45292849, -0.39365031,
       -0.40064735, -0.46513506, -0.48209563, -0.39635657, -0.44569517,
```

-0.4496465 , -0.43243444, -0.39406682, -0.40625773, -0.39826724,
 -0.46462545, -0.40782229, -0.42572527, -0.46599338, -0.42852596,
 -0.40082304, -0.38971614, -0.4628486 , -0.4219109 , -0.46365237,
 -0.39237271, -0.40320941, -0.42432754, -0.40309339, -0.40204058,
 -0.39044555, -0.43501511, -0.4324525 , -0.40503508, -0.40259674,
 -0.42956023, -0.42799201, -0.56257644, -0.38875652, -0.47585014,
 -0.3835507 , -0.4556408 , -0.406217 , -0.40385118, -0.39458774,
 -0.40122018, -0.40401747, -0.4443972 , -0.38320086, -0.3834212 ,
 -0.44945534, -0.4060406 , -0.38476589, -0.46817324, -0.38466101,
 -0.47610349, -0.49275615, -0.38344594, -0.40974845, -0.42395885,
 -0.42189161, -0.49261883, -0.48518689, -0.40901991, -0.39452123,
 -0.44785724, -0.40375287, -0.45749968, -0.40496115, -0.4260838 ,
 -0.41810669, -0.44560803, -0.38806155, -0.45523273, -0.38817405,
 -0.38177641, -0.39724261, -0.40279204, -0.46677259, -0.42162856,
 -0.4086809 , -0.43961327, -0.40404401, -0.45862325, -0.40473907,
 -0.41509113, -0.38081908, -0.39036169, -0.42441162, -0.52130968,
 -0.41557892, -0.40347146, -0.39318444, -0.38921027, -0.46342999,
 -0.39756969, -0.41357841, -0.38429305, -0.40297782, -0.40881293,
 -0.60259641, -0.44411547, -0.57052254, -0.50247903, -0.73212838,
 -0.64786135, -0.55623141, -0.45375541, -0.68754218, -0.67833274,
 -0.70698798, -0.63422967, -0.64031155, -0.78183101, -0.65477657,
 -0.67178443, -0.67084008, -0.68580391, -0.71446705, -0.64861428])

2 빅데이터와 금융자료 분석 기말대체과제

20249132 김형환

2.1 Question 1

- prob1_bank.csv 자료는 어느 포르투갈 은행의 정기예금 프로모션 전화 데이터이다. 이 데이터는 고객의 특징을 나타내는 특성 변수들과 고객이 정기예금에 가입했는지 여부를 나타내는 목표 변수로 구성되어 있다.

```
<특성변수>
age : 나이
job : 직업의 형태
marital : 결혼 상태
education : 학력
default : 신용 불이행 여부
balance : 은행 잔고
housing : 부동산 대출 여부
loan : 개인 대출 여부
contact : 연락 수단
month : 마지막으로 연락한 달

<목표변수>
y : 고객이 정기 예금에 가입했는지 여부
```

2.1.1 (1)

주어진 자료 중 범주형 변수 각각에 대해 적절한 전처리를 선택하고 진행하여라.

```
import numpy as np
import pandas as pd

bank = pd.read_csv('data/prob1_bank.csv')
bank.info() # 전체 11개 칼럼에 null값은 없으며, 범주형 9개 및 숫자형 2개

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          -----          ----- 
 0   age         4521 non-null   int64  
 1   job          4521 non-null   object  
 2   marital     4521 non-null   object  
 3   education   4521 non-null   object  
 4   default     4521 non-null   bool    
 5   balance     4521 non-null   int64  
 6   housing     4521 non-null   int64  
 7   loan         4521 non-null   int64  
 8   contact     4521 non-null   object  
 9   month        4521 non-null   object  
 10  y            4521 non-null   bool    
```

```
0    age        4521 non-null  int64
1    job        4521 non-null  object
2  marital     4521 non-null  object
3  education   4521 non-null  object
4  default     4521 non-null  object
5  balance     4521 non-null  int64
6  housing     4521 non-null  object
7  loan         4521 non-null  object
8  contact     4521 non-null  object
9  month        4521 non-null  object
10   y          4521 non-null  object

dtypes: int64(2), object(9)
memory usage: 388.6+ KB
```

```
bank.select_dtypes(include='object').nunique()
```

```
# 범주형 9개 중, 목적변수를 포함하여 4개는 '여부'에 대한 이진변수이며 나머지는 3~12개의 고유값
```

```
job           12
marital       3
education     4
default       2
housing       2
loan          2
contact       3
month         12
y              2

dtype: int64
```

```
bank['job'].value_counts()
```

```
# 직업의 경우, 12개의 범주로 구성됨. 특별히 한 값에 치중되는 모습도 보이지 않고,
# 순서가 없으므로 원-핫 인코딩으로 처리 예정
```

```
job
management      969
blue-collar     946
technician      768
admin.          478
services         417
retired          230
```

```
self-employed      183
entrepreneur       168
unemployed        128
housemaid         112
student            84
unknown            38
Name: count, dtype: int64
```

```
bank['marital'].value_counts()
```

```
# 결혼 여부는 싱글/결혼/이혼 3진변수임. 순서가 없으므로 원-핫 인코딩으로 처리 예정
```

```
marital
married          2797
single            1196
divorced          528
Name: count, dtype: int64
```

```
bank['education'].value_counts()
```

```
# 학력에 대한 내용은 primary - secondary - tertiary 순이므로 1~3 라벨인코딩 처리 예정
```

```
# unknown은 학력수준이 낮을 가능성성이 높을 것으로 추정, 0으로 처리하여 하나의 응답값으로 간주함.
```

```
education
secondary         2306
tertiary          1350
primary           678
unknown           187
Name: count, dtype: int64
```

```
bank['contact'].value_counts()
```

```
# 연락방법에 대한 내용은 telephone - cellular 순으로 연락이 편리하므로 1~2 라벨인코딩 처리 예정
```

```
# unknown은 연락 편의성이 가장 떨어질 것으로 보임. 0으로 처리하여 라벨인코딩 처리 예정
```

```
contact
cellular          2896
unknown           1324
telephone         301
Name: count, dtype: int64
```

```
bank['month'].value_counts()  
# 달에 대한 내용으로, 1~12개월 순서에 따른 라벨인코딩 처리 예정
```

```
month  
may      1398  
jul       706  
aug       633  
jun       531  
nov       389  
apr       293  
feb       222  
jan       148  
oct        80  
sep        52  
mar        49  
dec        20  
Name: count, dtype: int64
```

```
# 범주형 변수 처리
```

```
# 1. job, marital -> One-Hot Encoding  
bank = pd.get_dummies(bank, columns=['job', 'marital'], drop_first=True)  
  
# 2. education, contact, month > Label Encoding  
edu_map = {'unknown': 0, 'primary': 1, 'secondary': 2, 'tertiary': 3}  
bank['education'] = bank['education'].map(edu_map)  
  
contact_map = {'unknown': 0, 'telephone': 1, 'cellular': 2}  
bank['contact'] = bank['contact'].map(contact_map)  
  
month_order = ['jan', 'feb', 'mar', 'apr', 'may', 'jun',  
               'jul', 'aug', 'sep', 'oct', 'nov', 'dec']  
month_map = {month: i for i, month in enumerate(month_order)}  
bank['month'] = bank['month'].map(month_map)  
  
# 3. default, housing, loan, y > Label Encoding (binary, yes=1 / no=0)  
binary_cols = ['default', 'housing', 'loan', 'y']  
for col in binary_cols: bank[col] = bank[col].map({'yes': 1, 'no': 0})
```

```
print(bank.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   age               4521 non-null    int64  
 1   education         4521 non-null    int64  
 2   default           4521 non-null    int64  
 3   balance           4521 non-null    int64  
 4   housing           4521 non-null    int64  
 5   loan              4521 non-null    int64  
 6   contact           4521 non-null    int64  
 7   month             4521 non-null    int64  
 8   y                 4521 non-null    int64  
 9   job_blue-collar  4521 non-null    bool   
 10  job_entrepreneur 4521 non-null    bool   
 11  job_housemaid   4521 non-null    bool   
 12  job_management   4521 non-null    bool   
 13  job_retired      4521 non-null    bool   
 14  job_self-employed 4521 non-null    bool   
 15  job_services     4521 non-null    bool   
 16  job_student       4521 non-null    bool   
 17  job_technician   4521 non-null    bool   
 18  job_unemployed   4521 non-null    bool   
 19  job_unknown       4521 non-null    bool   
 20  marital_married  4521 non-null    bool   
 21  marital_single   4521 non-null    bool   

dtypes: bool(13), int64(9)
memory usage: 375.4 KB
None
```

2.1.2 (2)

주어진 자료 중 수치형 변수 각각에 대해 적절한 전처리를 선택하고 진행하여라.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```

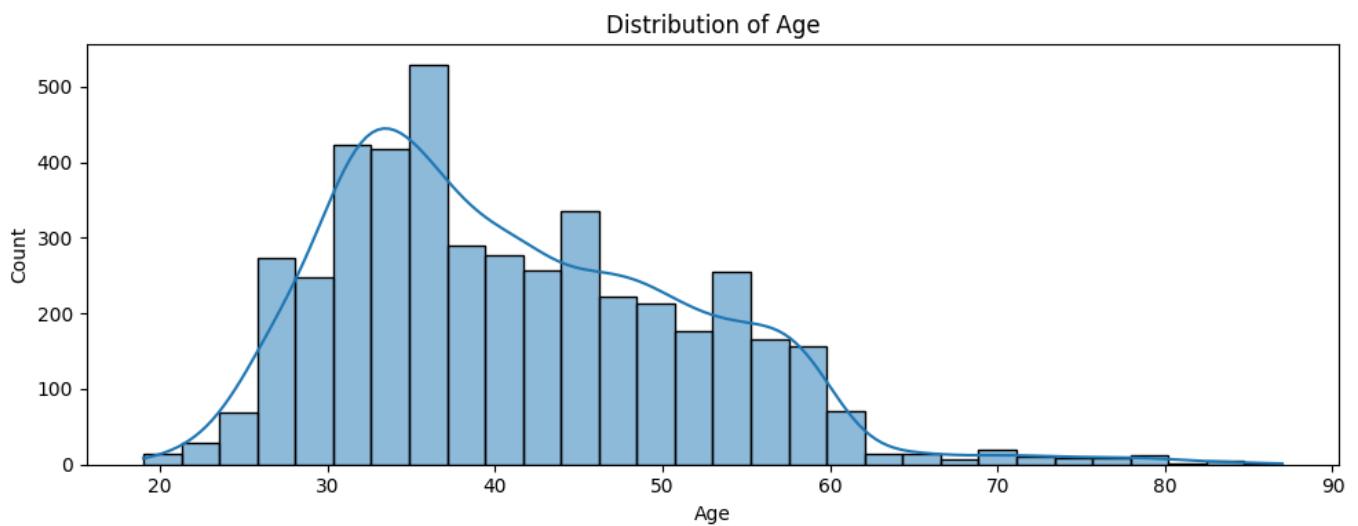
# 1. age 분포
plt.figure(figsize=(10, 4))
sns.histplot(bank['age'], bins=30, kde=True)
plt.title("Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Count")
plt.tight_layout()
plt.show()

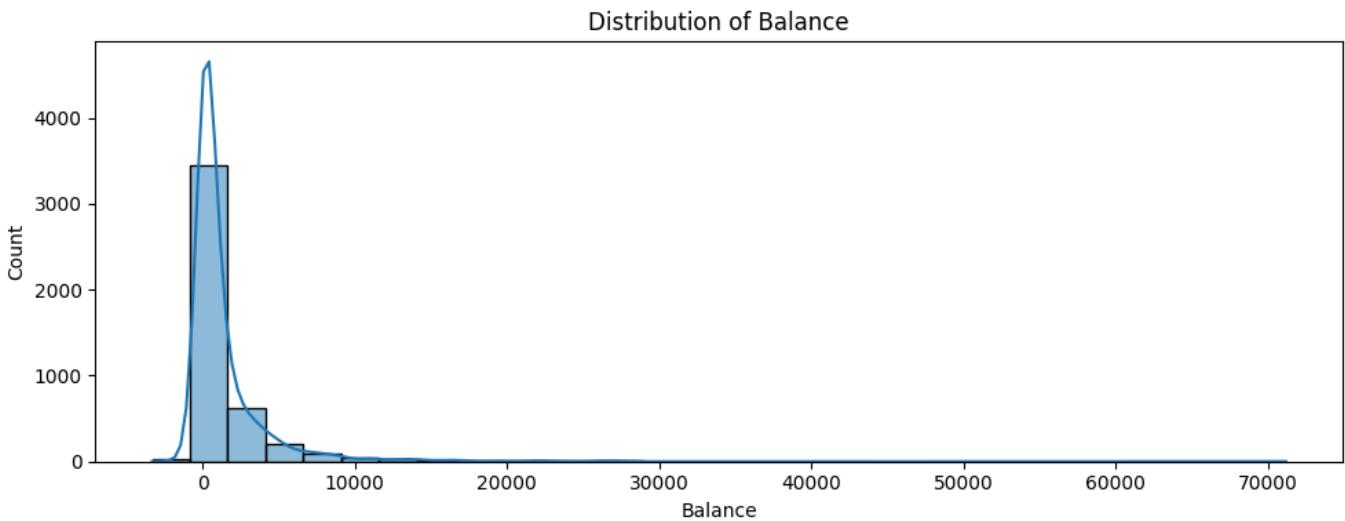
```

```

# 2. balance 분포
plt.figure(figsize=(10, 4))
sns.histplot(bank['balance'], bins=30, kde=True)
plt.title("Distribution of Balance")
plt.xlabel("Balance")
plt.ylabel("Count")
plt.tight_layout()
plt.show()

```





```
bank['age'].describe()
```

age는 오른쪽 skew가 미세하게 있는 정규분포와 가까운 형태. 표준화만 진행

```
count      4521.000000
mean       41.170095
std        10.576211
min        19.000000
25%       33.000000
50%       39.000000
75%       49.000000
max       87.000000
Name: age, dtype: float64
```

```
bank['balance'].describe()
```

balance는 음수도 존재하고, 값이 매우 극단적으로 치우쳐져 있는 형태. log변환 및 표준화 진행

이후 LOF 방식으로 두 수치형변수에 대한 이상치 탐지, 약 1% 수준의 이상치 제거 예정

```
count      4521.000000
mean       1422.657819
std        3009.638142
min       -3313.000000
25%       69.000000
50%       444.000000
75%      1480.000000
max      71188.000000
Name: balance, dtype: float64
```

```

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
import numpy as np

# 1. balance 로그변환 (음수 대비)
min_bal = bank['balance'].min()
bank['log_balance'] = np.log1p(bank['balance'] - min_bal + 1)

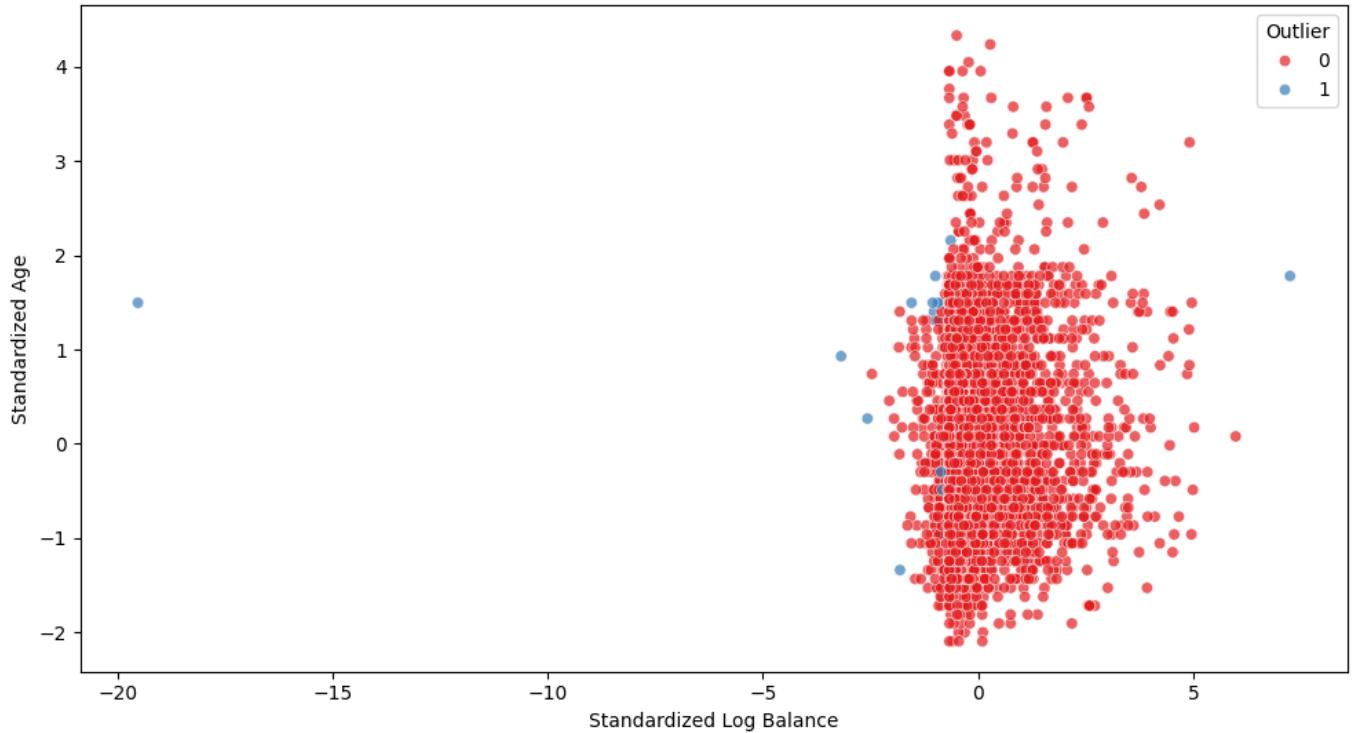
# 2. balance + age 표준화
scaler = StandardScaler()
bank[['std_log_balance', 'std_age']] = scaler.fit_transform(bank[['log_balance', 'age']])

# 3. LOF 이상치 탐지 (다변량: log_balance + age)
# 적절한 파라미터 조정으로 balance의 최소, 최대값을 효과적으로 제거
X_scaled = bank[['std_log_balance', 'std_age']]
lof = LocalOutlierFactor(n_neighbors=30, contamination=0.01)
bank['is_outlier'] = (lof.fit_predict(X_scaled) == -1).astype(int)

# 4. 이상치 시각화 (scatter plot)
plt.figure(figsize=(10, 6))
sns.scatterplot(data=bank, x='std_log_balance', y='std_age',
                 hue='is_outlier', palette='Set1', alpha=0.7)
plt.title("Scatter Plot of Standardized Log Balance vs. Age\n(LOF Outlier Detection)")
plt.xlabel("Standardized Log Balance")
plt.ylabel("Standardized Age")
plt.legend(title="Outlier")
plt.tight_layout()
plt.show()

```

Scatter Plot of Standardized Log Balance vs. Age
(LOF Outlier Detection)



이상치 1%가 제거된 것을 확인할 수 있음

```
bank_clean = bank[bank['is_outlier'] == 0].copy()
```

```
bank_clean
```

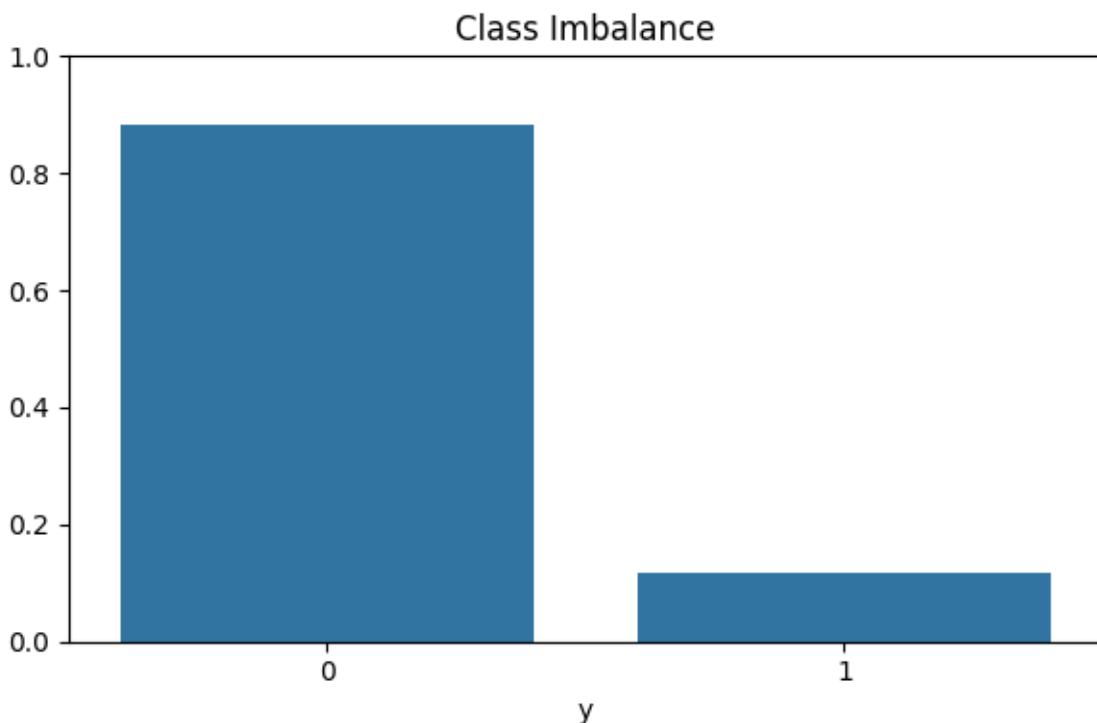
	age	education	default	balance	housing	loan	contact	month	y	job_blue-collar	...	job_stude
0	30	1	0	1787	0	0	2	9	0	False	...	False
1	33	2	0	4789	1	1	2	4	0	False	...	False
2	35	3	0	1350	1	0	2	3	0	False	...	False
3	30	3	0	1476	1	1	0	5	0	False	...	False
4	59	2	0	0	1	0	0	4	0	True	...	False
...
4515	32	2	0	473	1	0	2	6	0	False	...	False
4516	33	2	0	-333	1	0	2	6	0	False	...	False
4518	57	2	0	295	0	0	2	7	0	False	...	False
4519	28	2	0	1137	0	0	2	1	0	True	...	False
4520	44	3	0	1136	1	1	2	3	0	False	...	False

2.1.3 (3)

주어진 자료에 클래스 불균형이 있는지 확인한 뒤, 이에 대한 적절한 전처리 방법을 선택하여 진행하여라.

```
# 약 9:1로 0(No)의 비율이 압도적으로 많음. 클래스불균형 존재
```

```
class_counts = bank_clean['y'].value_counts(normalize=True)
plt.figure(figsize=(6, 4))
sns.barplot(x=class_counts.index, y=class_counts.values, legend=False)
plt.title("Class Imbalance")
plt.xlabel("y")
plt.ylim(0, 1)
plt.tight_layout()
plt.show()
```

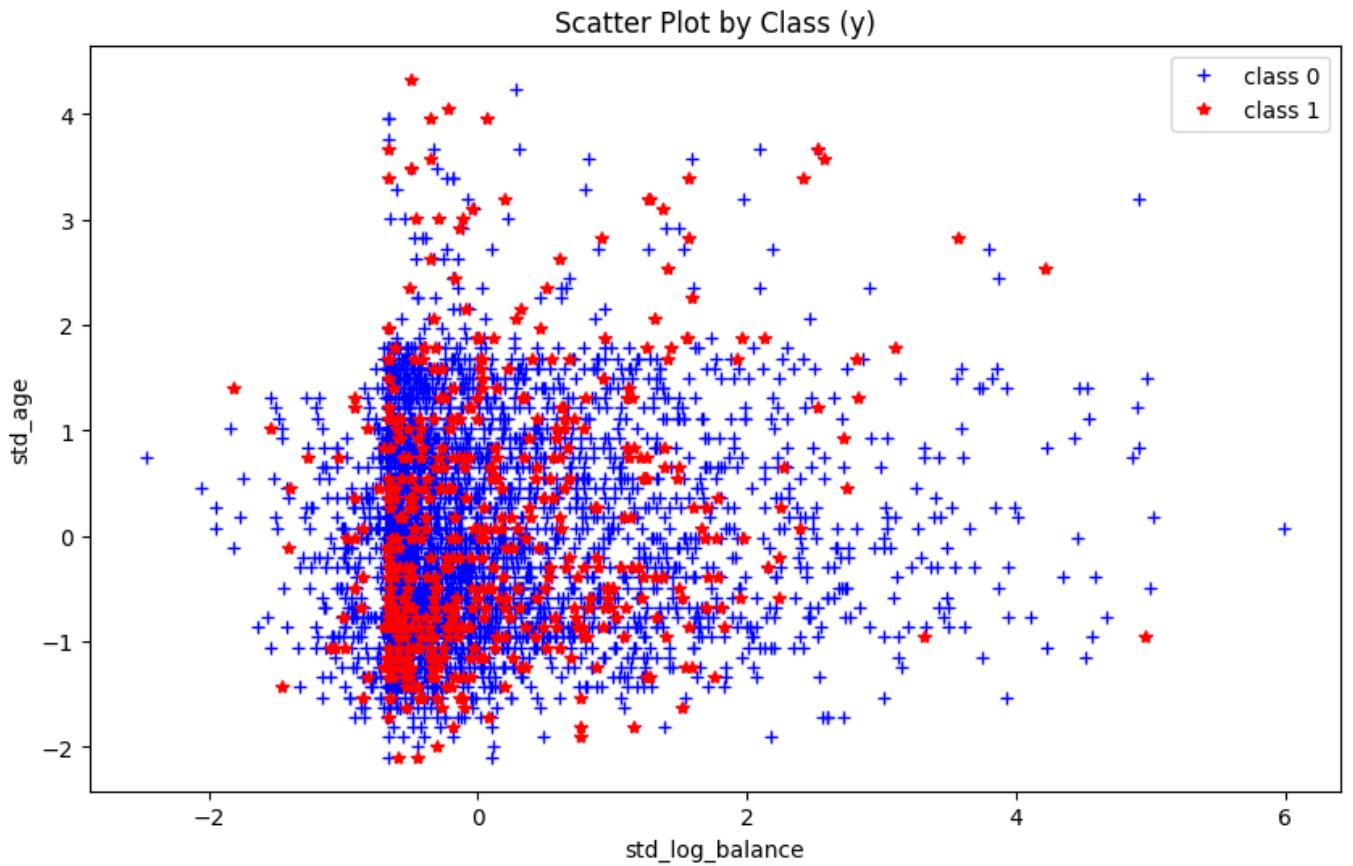


```
# 수치형 변수의 scatter plot상으로 특정 수치형 변수에 따른 치우침은 관측되지 않음.
```

```
X = bank_clean[['std_log_balance', 'std_age']]
y = bank_clean['y']
plt.figure(figsize=(10, 6))

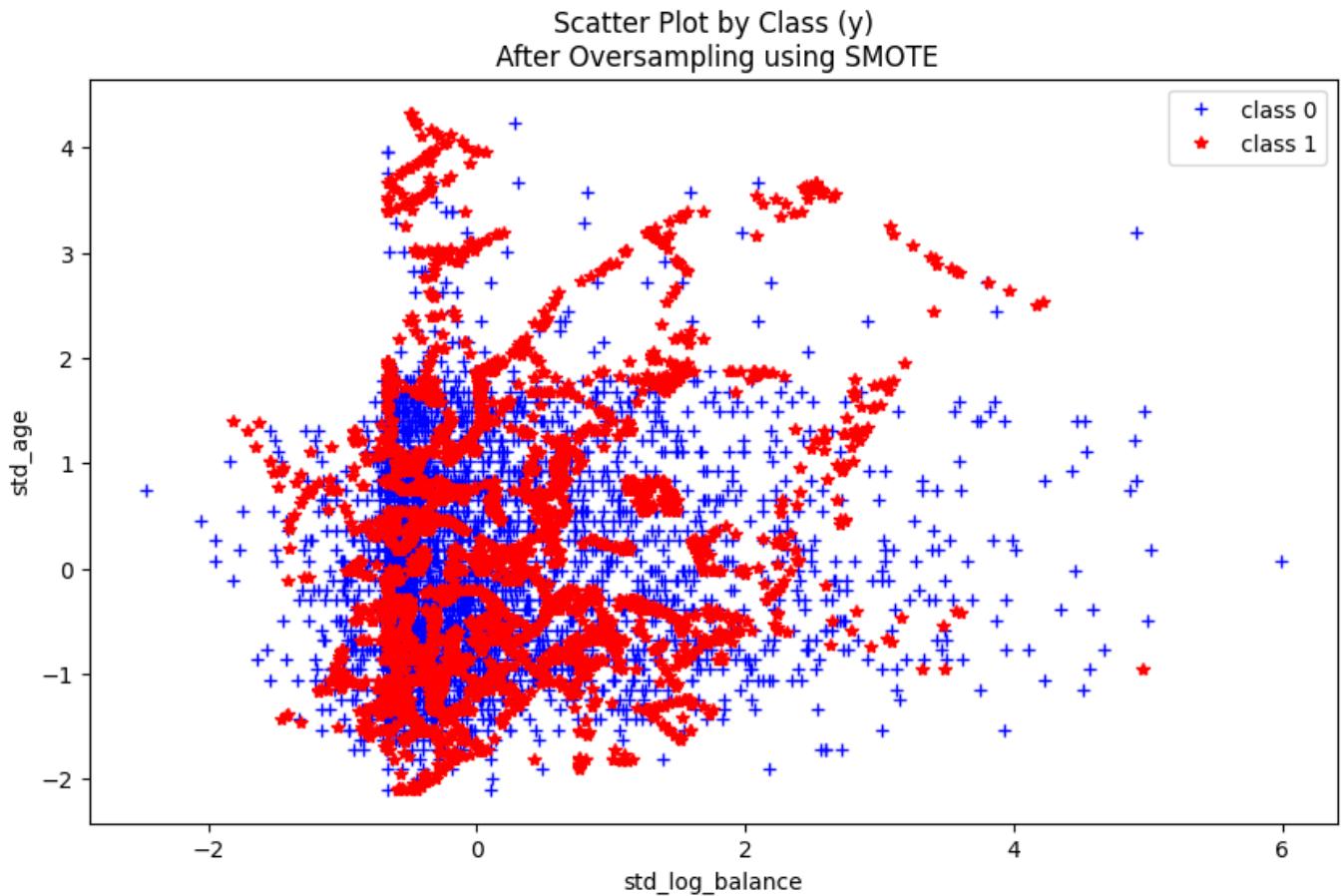
plt.plot(X.loc[y == 0, 'std_log_balance'], X.loc[y == 0, 'std_age'], 'b+', label="class 0")
plt.plot(X.loc[y == 1, 'std_log_balance'], X.loc[y == 1, 'std_age'], 'r*', label="class 1")
plt.legend()

plt.xlabel("std_log_balance")
plt.ylabel("std_age")
plt.title("Scatter Plot by Class (y)")
plt.show()
```



```
# 표본의 수가 적은 편이므로 oversampling 진행
# 데이터의 구조가 그다지 복잡하지 않아 SMOTE 알고리즘을 적용하여 처리
from imblearn.over_sampling import SMOTE

oversample1 = SMOTE()
OX, Oy = oversample1.fit_resample(X, y)
plt.figure(figsize=(10, 6))
plt.plot(OX.loc[Oy == 0, 'std_log_balance'], OX.loc[Oy == 0, 'std_age'], 'b+', label="class 0")
plt.plot(OX.loc[Oy == 1, 'std_log_balance'], OX.loc[Oy == 1, 'std_age'], 'r*', label="class 1")
plt.legend()
plt.xlabel("std_log_balance")
plt.ylabel("std_age")
plt.title("Scatter Plot by Class (y)\n After Oversampling using SMOTE")
plt.show()
```



2.2 Question 2

2. `prob2_card.csv` 자료는 어느 신용카드 회사의 고객 데이터로, 신용카드 사용 형태를 나타내는 여러 특성 변수들로 구성되어 있다.

CUST_ID :	신용카드 사용자 ID
BALANCE :	구매 계좌 잔액
BALANCE_FREQUENCY :	구매 계좌 잔액이 업데이트 되는 빈도 지수로, 0(자주 업데이트 되지 않음)~1(자주 업데이트 됨) 사이의 값을 가짐.
PURCHASES :	구매 계좌로부터의 구매액
PURCHASES_FREQUENCY :	구매 빈도 지수로, 0(자주 구매하지 않음)~1(자주 구매함) 사이의 값을 가짐.
PURCHASES_TRX :	구매 거래 건수

2.2.1 (1)

주어진 자료에 K평균 Clustering 알고리즘을 적용하여, 적절한 군집을 생성하여라.

```
df = pd.read_csv('data/prob2_card.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
```

```
Data columns (total 6 columns):
```

#	Column	Non-Null Count	Dtype
0	CUST_ID	8950 non-null	object
1	BALANCE	8950 non-null	float64
2	BALANCE_FREQUENCY	8950 non-null	float64
3	PURCHASES	8950 non-null	float64
4	PURCHASES_FREQUENCY	8950 non-null	float64
5	PURCHASES_TRX	8950 non-null	int64

dtypes: float64(4), int64(1), object(1)
memory usage: 419.7+ KB

```
df.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	PURCHASES_FREQUENCY	PURCHASES_TRX
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	1003.204834	0.490351	14.709832
std	2081.531879	0.236904	2136.634782	0.401371	24.857649
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	128.281915	0.888889	39.635000	0.083333	1.000000
50%	873.385231	1.000000	361.280000	0.500000	7.000000
75%	2054.140036	1.000000	1110.130000	0.916667	17.000000
max	19043.138560	1.000000	49039.570000	1.000000	358.000000

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# CUST_ID는 클러스터링에 사용하지 않으므로 제거
X = df.drop(columns=["CUST_ID"])

# 표준화
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# 최적 K 탐색
inertia_list = []
silhouette_list = []
K_range = range(2, 11)
```

```

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)
    inertia_list.append(kmeans.inertia_)
    silhouette_list.append(silhouette_score(X_scaled, labels))

best_k = K_range[silhouette_list.index(max(silhouette_list))]

# K평균 군집화 결과 : 6개의 군집으로 분류되었으며, 갯수는 32개~3200개로 천차만별
kmeans_final = KMeans(n_clusters=best_k, random_state=42, n_init=10)
scaled_df["KMeans_Label"] = kmeans_final.fit_predict(X_scaled)

summary_table_kmean = scaled_df.groupby("KMeans_Label").mean()
summary_table_kmean["Count"] = scaled_df.groupby("KMeans_Label").size()
summary_table_kmean

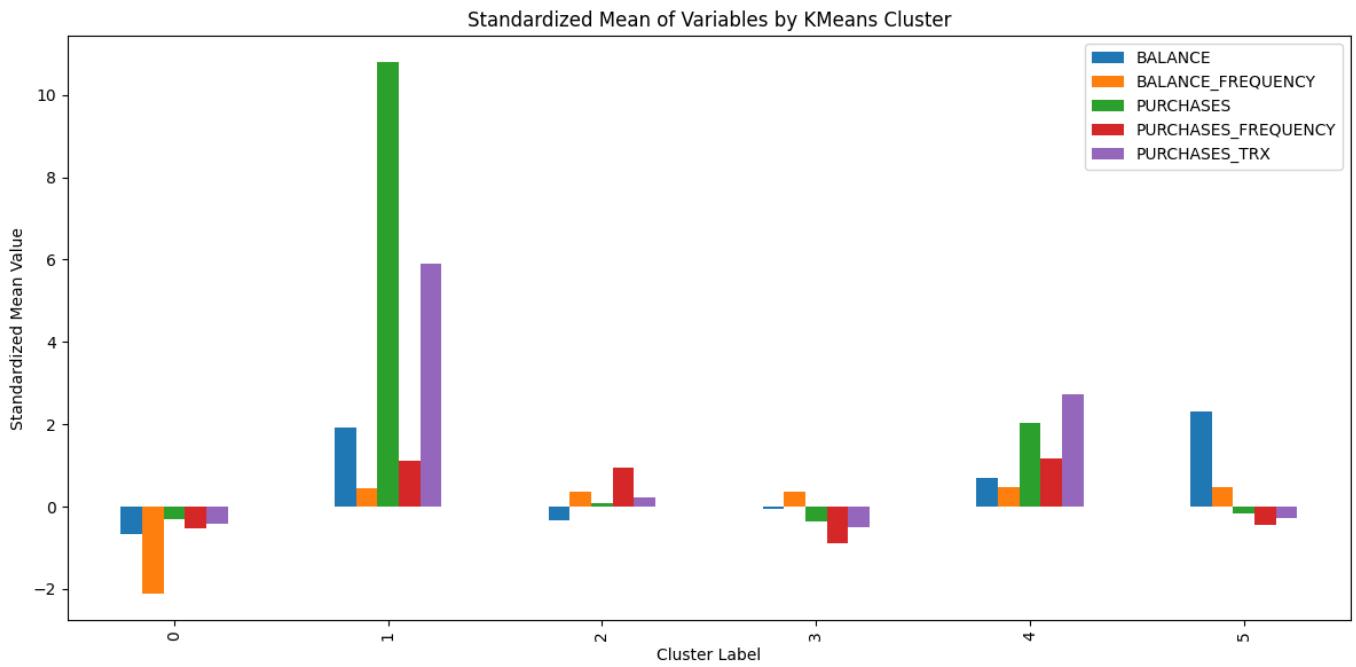
```

KMeans_Label	BALANCE	BALANCE_FREQUENCY	PURCHASES	PURCHASES_FREQUENCY	PUI
0	-0.679901	-2.105347	-0.305189	-0.528487	-0.4
1	1.906921	0.444930	10.782703	1.119309	5.89
2	-0.331950	0.368356	0.087095	0.960328	0.22
3	-0.068576	0.366930	-0.369541	-0.895659	-0.5
4	0.696282	0.475045	2.042474	1.179006	2.71
5	2.320551	0.483046	-0.159336	-0.433551	-0.2

```

# K평균 군집화 시각화
summary_table_kmean.drop(columns="Count").plot.bar(figsize=(12, 6))
plt.title("Standardized Mean of Variables by KMeans Cluster")
plt.xlabel("Cluster Label")
plt.ylabel("Standardized Mean Value")
plt.tight_layout()

```



2.2.2 (2)

주어진 자료에 DBSCAN Clustering 알고리즘을 적용하여, 적절한 군집을 생성하여라.

```

from sklearn.cluster import DBSCAN

# DBSCAN 분류 : eps 및 min_samples는 여러번 반복을 통해 최적의 조합을 도출하였음.
# 기준 : 분류가 너무 많거나 적지 않도록(3~6개), 너무 숫자가 적은 분류가 없도록
dbscan = DBSCAN(eps=0.6, min_samples=4)
labels = dbscan.fit_predict(X_scaled)

# 라벨을 데이터프레임에 추가
scaled_df["DBSCAN_Label"] = labels

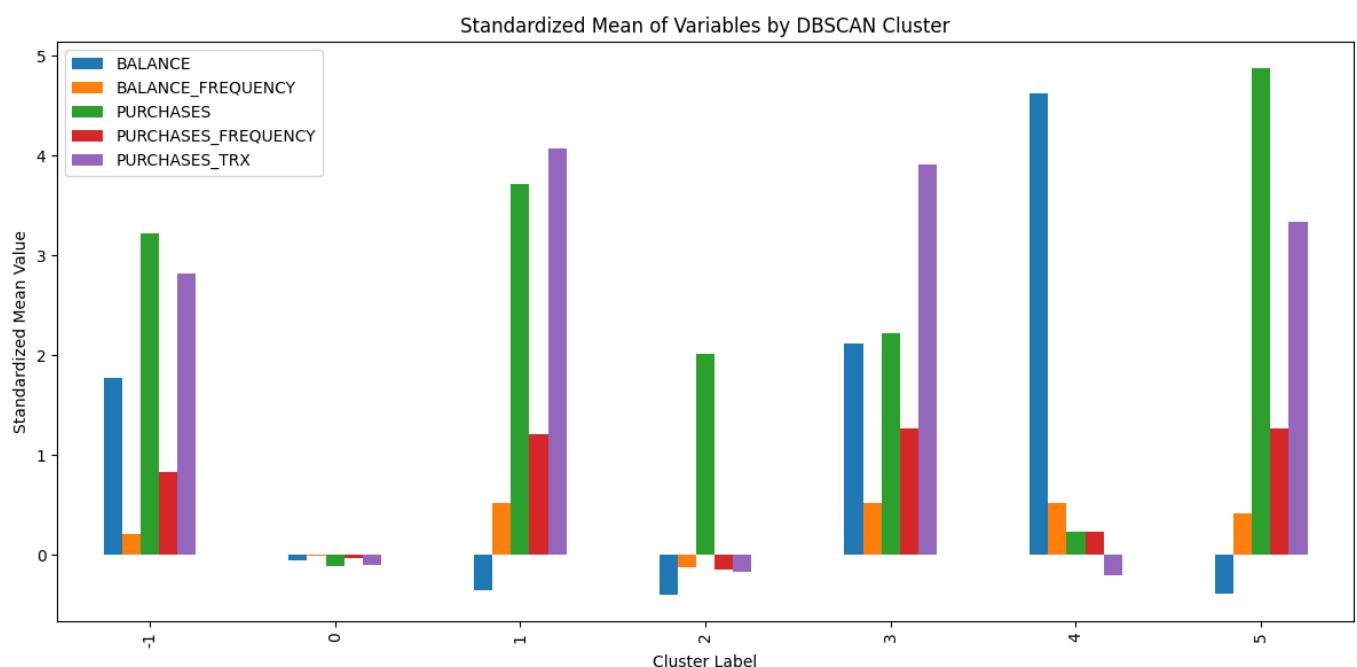
# DBSCAN 군집화 결과 : 잡음(-1)이 약 2% 포함되어있으며, 총 5개의 군집으로 분류(177개~6200개)
summary_table_db = scaled_df.drop(columns="KMeans_Label").groupby("DBSCAN_Label").mean()
summary_table_db["Count"] = scaled_df.groupby("DBSCAN_Label").size()
summary_table_db

```

DBSCAN_Label	BALANCE	BALANCE_FREQUENCY	PURCHASES	PURCHASES_FREQUENCY	PURCHASES_TRX
-1	1.776034	0.211613	3.221764	0.832535	2.8
0	-0.056781	-0.007807	-0.107234	-0.028259	-0.
1	-0.355658	0.518084	3.714917	1.207553	4.0

DBSCAN_Label	BALANCE	BALANCE_FREQUENCY	PURCHASES	PURCHASES_FREQUENCY	PURCHASES_TRX
2	-0.397981	-0.121515	2.012334	-0.148985	-0.384296
3	2.124764	0.518084	2.221881	1.269843	3.923668
4	4.623668	0.518084	0.235086	0.231676	-0.384296
5	-0.384296	0.422144	4.876114	1.269843	3.923668

```
summary_table_db.drop(columns="Count").plot.bar(figsize=(12, 6))
plt.title("Standardized Mean of Variables by DBSCAN Cluster")
plt.xlabel("Cluster Label")
plt.ylabel("Standardized Mean Value")
plt.tight_layout()
```



2.2.3 (3)

(1)과 (2)의 두 군집 분석 결과를 비교하고, 더 타당한 모델을 선택하여라.

K평균 vs DBSCAN 군집화 결과 비교

항목	K평균	DBSCAN
클러스터 수	6개	6개 (-1 포함)
클러스터 구성	최소 32, 최대 3253명	최소 4, 최대 8657
이상치 처리	없음	-1로 261명 처리
군집별 특성	분류기준 명확하	분류기준 모호, 대부분 하나로 분류

결론 : 이 데이터에서는 **K평균**이 더 타당한 군집화 방법

2.2.4 (4)

(3)에서 선택된 최종 모델로 생성한 군집들의 고객 특성을 분석하여라.

K평균 군집화 군집별 특성

0 : 낮은 자산, 낮은 구매액, 낮은 구매횟수 -> 하위 고객군 (약 17.5%)

1 : 높은 자산, 높은 구매액, 높은 구매횟수 -> 부유하고 이용량 많은 VIP 고객군 (약 0.5%)

2 : 평균이하 자산, 평균적인 구매액, 평균이상 구매횟수 -> 일반 고객군 중 상위 이용고객 (약 35%)

3 : 평균적인 자산, 평균이하 구매액, 평균이하 구매횟수 -> 일반 고객군 중 하위 이용고객 (약 33%)

4 : 평균적인 자산, 높은 구매액, 높은 구매횟수 -> 평균적이나 카드 사용량이 많은 우량 고객군 (약 5%)

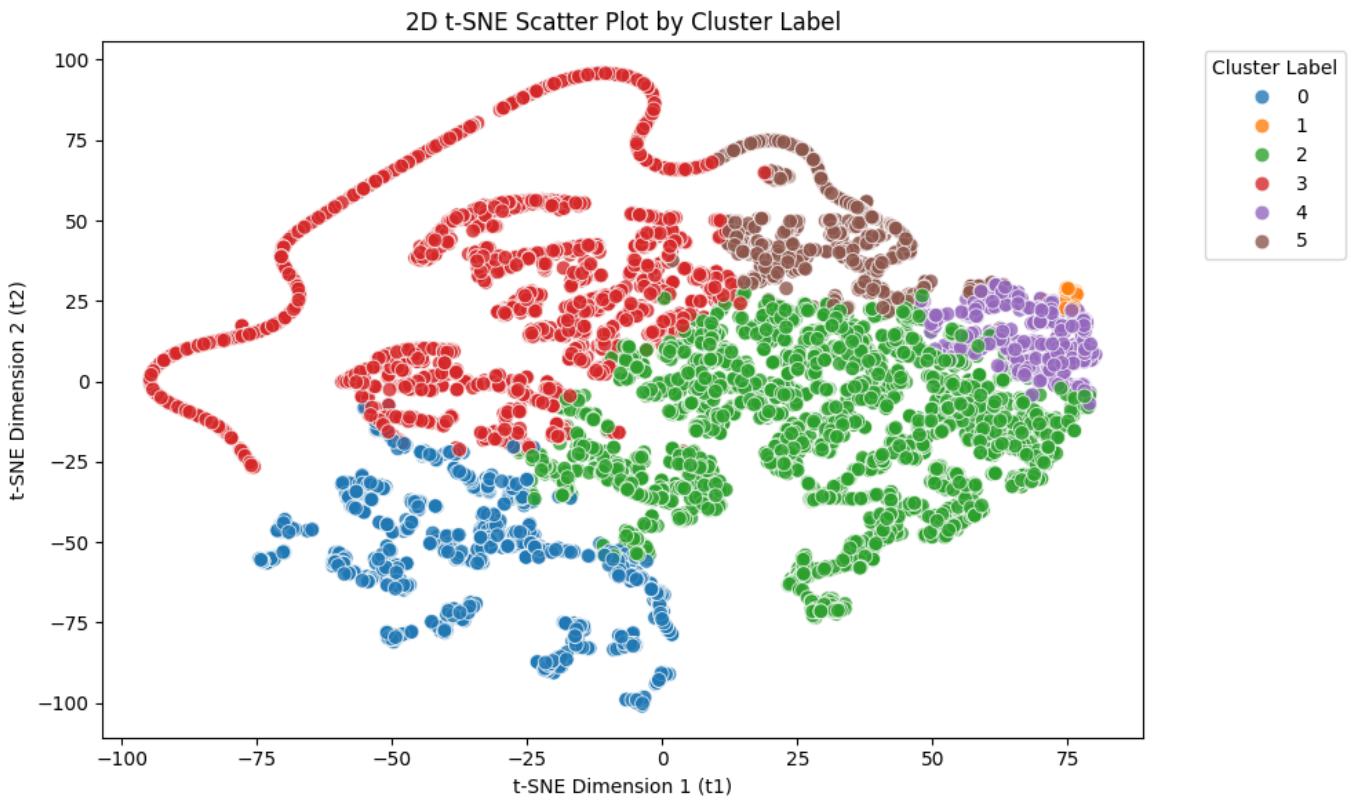
5 : 높은 자산, 낮은 구매액, 낮은 구매횟수 -> 부유하나 카드를 이용하지 않는 잠재 고객군 (약 9%)

2.2.5 (5)

t-SNE 알고리즘을 적용하여 주어진 자료를 2차원으로 축소하여라. 그 결과를, (3)에서 선택한 모델의 군집 레이블에 따라 점의 색상이 다르게 표현된 2차원 산점도로 시각화하여라.

```
from sklearn.manifold import TSNE
tsne = TSNE( n_components=2 )
df2dim = tsne.fit_transform( X_scaled )
df2dim = pd.DataFrame( df2dim, columns=['t1','t2'] )
df2dim['Labels' ] = scaled_df["KMeans_Label"]

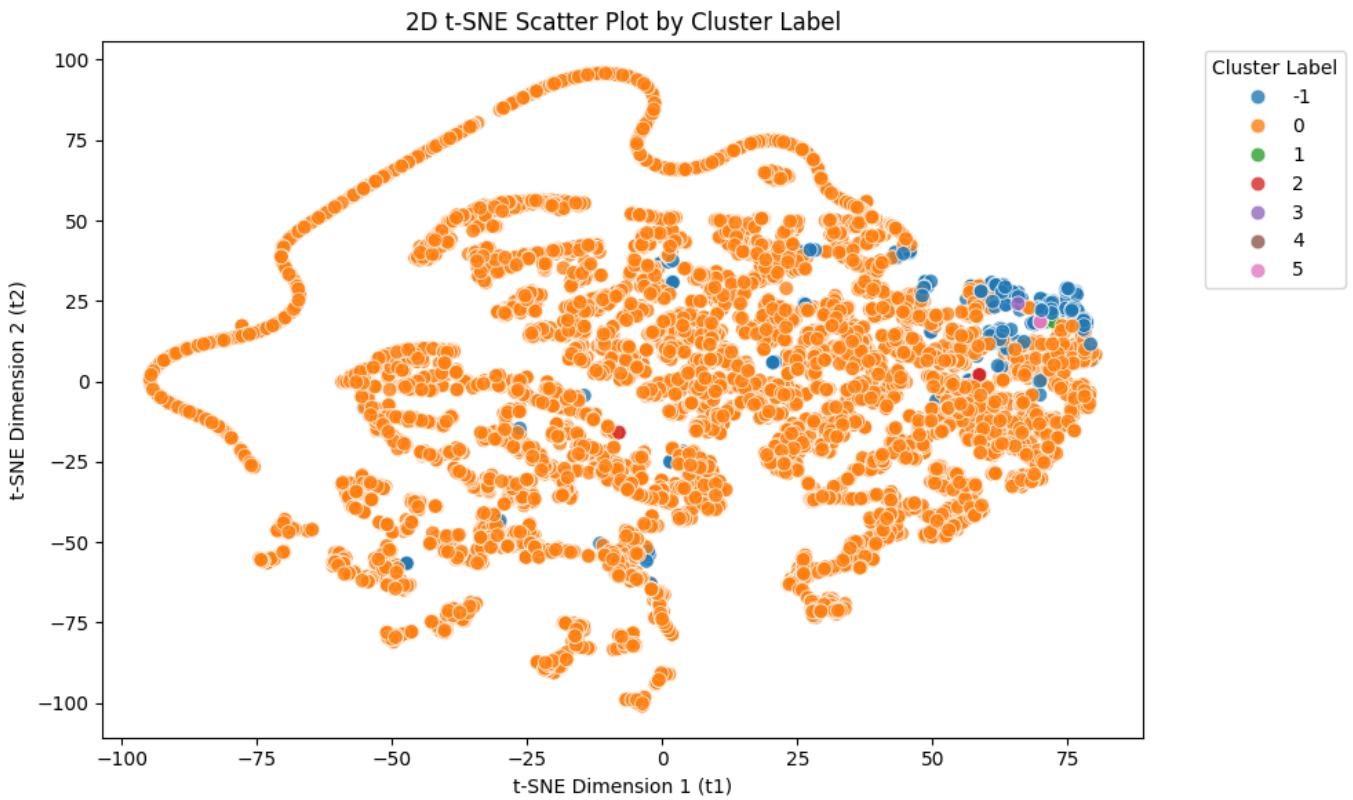
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df2dim, x="t1", y="t2", hue="Labels", palette="tab10", s=60, alpha=0.8)
plt.title("2D t-SNE Scatter Plot by Cluster Label")
plt.xlabel("t-SNE Dimension 1 (t1)")
plt.ylabel("t-SNE Dimension 2 (t2)")
plt.legend(title="Cluster Label", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



```
# 참고 : DBSCAN 시각화 결과
```

```
df2dim['Labels'] = scaled_df["DBSCAN_Label"]
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df2dim, x="t1", y="t2", hue="Labels", palette="tab10", s=60, alpha=0.8)
plt.title("2D t-SNE Scatter Plot by Cluster Label")
plt.xlabel("t-SNE Dimension 1 (t1)")
plt.ylabel("t-SNE Dimension 2 (t2)")
plt.legend(title="Cluster Label", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



2.3 Question 3

3. 다음 데이터에서 특성변수인 X는 방의 개수, 목표변수인 Y는 주택 가격을 나타낸다.

ID	X	Y
1	3	1.25
2	1	1.2
3	2	1.3
4	4	1.5
5	?	1.4
6	?	1.3

2.3.1 (1)

XGBoost 알고리즘을 적용하여 트리를 생성한다고 할 때, 첫번째 트리의 첫 마디에서 최적의 분리기준이 무엇인지를 구하여라. 단, 결측이 아닌 4개의 관찰치(ID 1~ID 4)만 이용할 것. 제곱오차 손실함수를 적용하며, 모델 초기값 0는 0.5로 두고, 규제 하이퍼 파라미터는 0으로 설정할 것. 또한 계산 과정을 상세하게 서술할 것.

1. Gradient 및 Hessian 계산

- 손실함수: $(y_i - \hat{y}_i)^2$
- 예측값: $\hat{y}_i = 0.5$
- Gradient: $g_i = \hat{y}_i - y_i = 0.5 - y_i$
- Hessian: $h_i = 1$ (제곱오차 손실함수는 상수)

ID	y_i	g_i	h_i
1	1.25	-0.75	1
2	1.20	-0.70	1
3	1.30	-0.80	1
4	1.50	-1.00	1

2. Split 후보 및 Gain 계산

Gain 공식 ($\lambda = 0$):

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{(G_L + G_R)^2}{H_L + H_R} \right]$$

Split 1: $X \leq 1.5$

- Left: ID 2 $\rightarrow G_L = -0.70, H_L = 1$
- Right: ID 1, 3, 4 $\rightarrow G_R = -2.55, H_R = 3$

$$\text{Gain}_1 = \frac{1}{2} (0.49 + 2.1675 - 2.640625) = 0.0084$$

Split 2: $X \leq 2.5$

- Left: ID 2, 3 $\rightarrow G_L = -1.50, H_L = 2$
- Right: ID 1, 4 $\rightarrow G_R = -1.75, H_R = 2$

$$\text{Gain}_2 = \frac{1}{2} (1.125 + 1.53125 - 2.640625) = 0.0078$$

Split 3: $X \leq 3.5$

- Left: ID 1, 2, 3 $\rightarrow G_L = -2.25, H_L = 3$

- Right: ID 4 $\rightarrow G_R = -1.00, H_R = 1$

$$\text{Gain}_3 = \frac{1}{2} (1.6875 + 1 - 2.640625) = 0.0234$$

결론

Split 조건	Gain
$X \leq 1.5$	0.0084
$X \leq 2.5$	0.0078
$X \leq 3.5$	0.0234

최적 분리 기준: $X \leq 3.5$

2.3.2 (2)

XGBoost 알고리즘을 적용하여 트리를 생성한다고 할 때, 첫번째 트리의 첫 마디에서 X의 값이 결측인 경우는 왼쪽과 오른쪽 자식마디 중 어느 쪽으로 보내야 할지를 결정하여라.

위의 최적 분할 기준 $X \leq 3.5$ 에 따라, 아래 상황임.

- Left: ID 1, 2, 3 $\rightarrow G_L = -2.25, H_L = 3$
 - Right: ID 4 $\rightarrow G_R = -1.00, H_R = 1$
-

1. 결측이 ID 5인 경우 ($Y = 1.4$)

- $g_5 = 0.5 - 1.4 = -0.9, h_5 = 1$

Case A: 왼쪽으로 보낼 경우

- $G_L = -3.15, H_L = 4$
- $G_R = -1.00, H_R = 1$

$$\text{Gain}_{5,\text{left}} = \frac{1}{2} \left(\frac{(-3.15)^2}{4} + \frac{(-1)^2}{1} - \frac{(-4.15)^2}{5} \right) = 0.0181$$

Case B: 오른쪽으로 보낼 경우

- $G_L = -2.25, H_L = 3$

- $G_R = -1.9, H_R = 2$

$$\text{Gain}_{5,\text{right}} = \frac{1}{2} \left(\frac{(-2.25)^2}{3} + \frac{(-1.9)^2}{2} - \frac{(-4.15)^2}{5} \right) = 0.0240$$

결론: 결측값이 ID 5인 경우, 오른쪽이 더 유리함

2. 결측이 ID 6인 경우 ($Y = 1.3$)

- $g_6 = 0.5 - 1.3 = -0.8, h_6 = 1$

Case A: 왼쪽으로 보낼 경우

- $G_L = -3.05, H_L = 4$
- $G_R = -1.00, H_R = 1$

$$\text{Gain}_{6,\text{left}} = \frac{1}{2} \left(\frac{(-3.05)^2}{4} + \frac{(-1)^2}{1} - \frac{(-4.05)^2}{5} \right) = 0.0229$$

Case B: 오른쪽으로 보낼 경우

- $G_L = -2.25, H_L = 3$
- $G_R = -1.8, H_R = 2$

$$\text{Gain}_{6,\text{right}} = \frac{1}{2} \left(\frac{(-2.25)^2}{3} + \frac{(-1.8)^2}{2} - \frac{(-4.05)^2}{5} \right) = 0.0135$$

결론: ID 6은 왼쪽이 더 유리함

요약

결측 ID	Y 값	왼쪽 Gain	오른쪽 Gain	더 나은 방향
ID 5	1.4	0.0181	0.0240	오른쪽
ID 6	1.3	0.0229	0.0135	왼쪽

빅데이터와 금융자료분석 프로젝트 (Team 4)

XGboost 알고리즘을 활용한 은행 대출의 부도 여부 예측 모델 구축

강상묵(20259013) 김형환(20249132) 유석호(20249264) 이현준(20249349) 최영서(20249430) 최재필(20249433)

1. 프로젝트 개요

본 프로젝트는 여러 데이터 전처리 기법(결측치, 이상치, 특성공학 등)과 머신러닝 알고리즘(이상치 분류, 차원축소, XGBoost 등)을 실제 금융데이터에 적용해보고 시사점을 도출하기 위해 작성되었습니다.

이를 위해 미국 Lending Club의 P2P 대출 데이터를 사용하였으며, 전반적인 워크플로우는 아래와 같습니다.

1. 데이터의 구조, 특성 파악 (EDA)
2. 데이터의 전처리 (특성에 따른 칼럼 가공, 문자형 변수 처리, 결측치 및 이상치 처리, 변수 선택)
3. XGBoost 알고리즘을 이용한 대출 연체여부 예측 모델 구축 및 평가
 - 샘플링, 모델 튜닝, 성과 평가, 변수 중요도 분석(SHAP), Cat/LightGBM 등 다른 모델과 비교

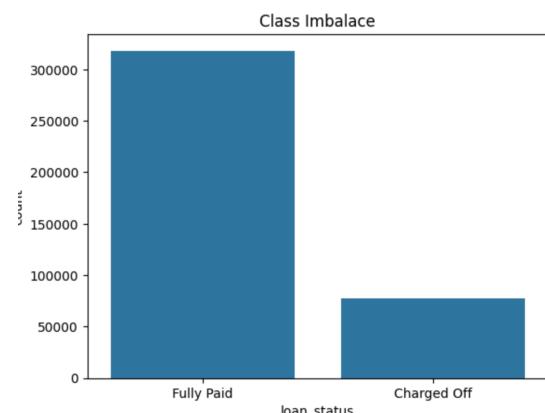
2. 데이터의 구조, 특성 (EDA)

데이터의 수집, 기본구조

미국 소재의 P2P 대출 전문은행인 Lending Club의 '07~'20년 대출 데이터를 사용하였습니다. (출처 : Kaggle)

약 40만개의 데이터로, 목적변수인 대출상태를 포함해 전체 27개의 칼럼(수치형 12 + 문자형 15)으로 이루어져 있으며, 목적변수는 정상(상환, Fully paid) 및 부도(연체, Charged off)로 이진분류 문제입니다.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   loan_amnt        396030 non-null   float64
 1   term             396030 non-null   object 
 2   int_rate          396030 non-null   float64
 3   installment       396030 non-null   float64
 4   grade            396030 non-null   object 
 5   sub_grade         396030 non-null   object 
 6   emp_title         373103 non-null   object 
 7   emp_length        377729 non-null   object 
 8   home_ownership    396030 non-null   object 
 9   annual_inc        396030 non-null   float64
 10  education_status 396030 non-null   object 
 11  issue_d          396030 non-null   object 
 12  loan_status       396030 non-null   object 
 13  purpose           396030 non-null   object 
 14  title             394274 non-null   object 
 15  dti               396030 non-null   float64
 16  earliest_cr_line 396030 non-null   object 
 17  next_pmt_dt      396030 non-null   float64
 18  pub_rec           396030 non-null   float64
 19  revol_bal         396030 non-null   float64
 20  revol_util        395754 non-null   float64
 21  total_acc         396030 non-null   float64
 22  initial_list_status 396030 non-null   object 
 23  application_type 396030 non-null   object 
 24  mort_acc          358830 non-null   float64
 25  total_rev_hi_risk 395495 non-null   float64
 26  address           396030 non-null   object 
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

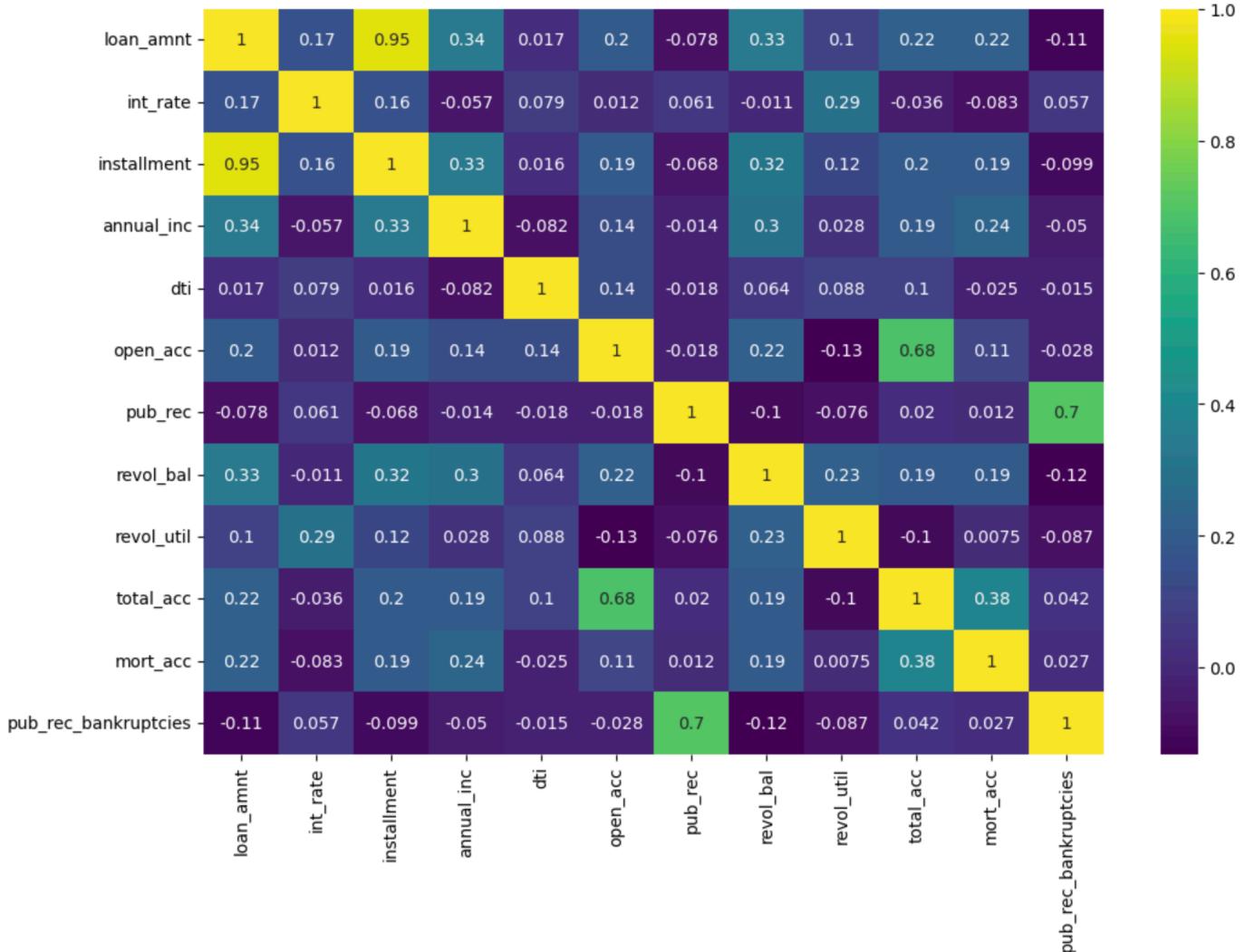


데이터의 특성

데이터의 각 칼럼별 특징을 알아보고, 적절한 전처리 방법을 탐색해보았습니다.

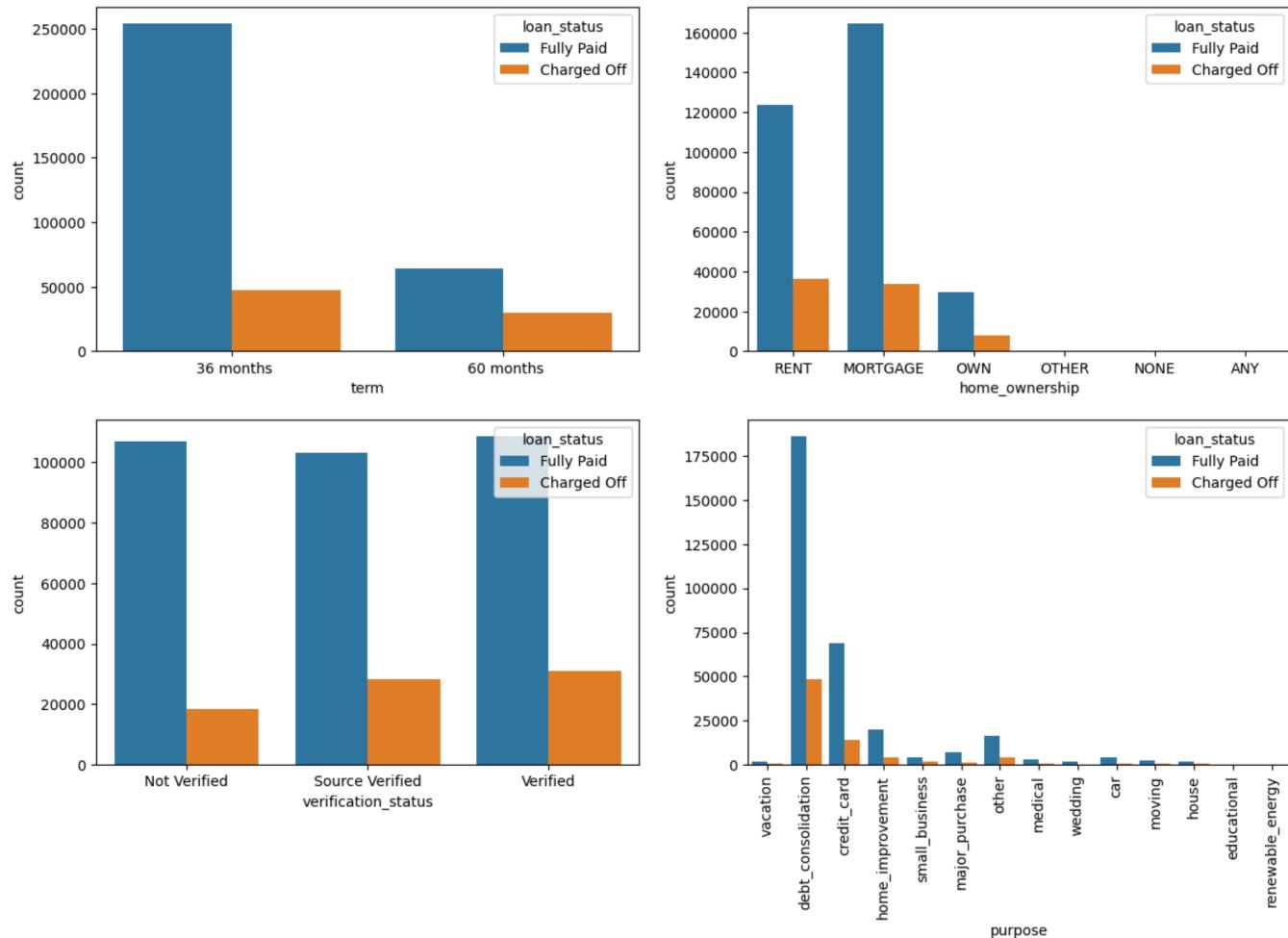
먼저 수치형 변수입니다. 결측치 및 이상치 처리는 별도 진행 예정으로 따로 다루지 않겠습니다.

상관관계 행렬을 **Heatmap**으로 살펴보았습니다. 대체적으로 변수들 간 상관관계가 미미하였으며, 일부 상관계수가 높은 변수들은 변수선택 과정에서 제외하는 등 별도의 전처리 과정을 통해 다중공선성 문제를 해결할 계획입니다.

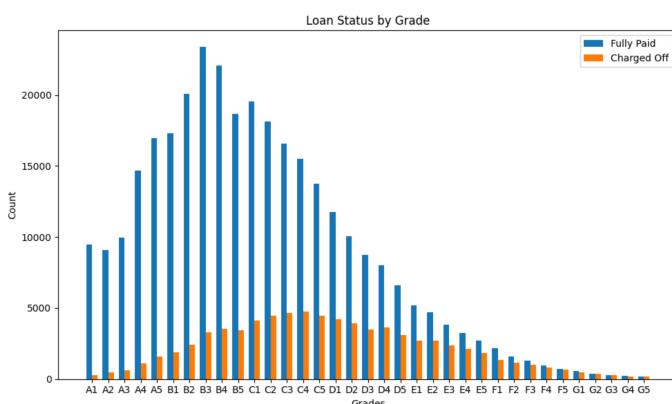


다음으로 문자형 변수입니다. 목적변수와 관련이 있는 것으로 보이는 주요 예시만 살펴보겠습니다.

먼저, 대출기간(term), 집보유형태(home_ownership), 대출목적(purpose)이 영향을 미치는 것으로 추정됩니다.



다음으로, 신용등급(A1~G5)에 따라 부도율이 높아지는 추이를 보였으며, 문자형 변수들 중 일부는 고유값이 너무 많아 분석에서 제외하는 것이 효과적일 것으로 보입니다.



term	2
grade	7
sub_grade	35
emp_title	173105
emp_length	11
home_ownership	6
verification_status	3
issue_d	115
loan_status	2
purpose	14
title	48816
earliest_cr_line	684
initial_list_status	2
application_type	3
address	393700
dtype: int64	

3. 데이터의 전처리

데이터의 전처리는 아래의 과정으로 실시하였습니다.

1. 분석에 적합하도록 칼럼 변환 및 통합, 제거

- 변환/통합 : 주소(address)는 우편번호(zip_code)만 추출하고 제거, 대출기간(term, 36month 등)은 수치형으로 변환, 집 소유여부(home_ownership)의 극소수값들은 Other로 통합
- 불필요한 noise 방지를 위해 100개 이상의 고유값을 가진 칼럼 제거 : 직업(title), 직업글자수(emp_title), 발행일(issue_d), 최초연도(earliest_cr_line)
- 다른 변수와 중복되거나 추론 가능한 칼럼 제거 : 신용점수-대분류(grade), 근속연수(emp_length)

2. 문자형 변수 처리 : 순서가 있거나 이진변수인 경우 라벨인코딩, 단순 점주인 경우 원핫인코딩 적용

- 라벨인코딩 : 목적변수(이진), 신용점수(순서 존재) / 원핫인코딩 : 이외의 문자형 변수

3. 수치형 변수의 결측치 및 이상치 처리 : 중간값 처리 및 1% 이상치 제거

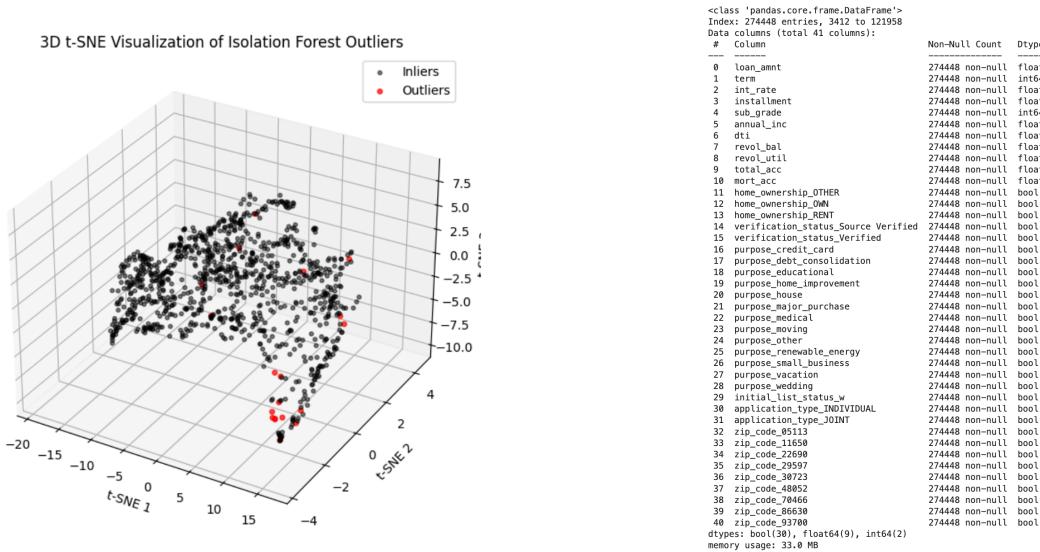
- 결측치 : 변수간 상관관계가 미미하고, 이후 Boruta를 적용 예정이므로 예측형 모델보다는 중간값을 채택
- 이상치 : 고차원, 많은 샘플(약 40만)을 고려, 분포에 대한 가정이 불필요한 Isolation forest 기법 채택

4. 변수 선택을 통해 분석에 적합한 최종 데이터 가공 : Boruta 알고리즘 적용

- 일부 변수간 상관관계가 존재하는 점을 고려, 최적의 변수 조합을 찾고자 Boruta 알고리즘 채택
- 원핫인코딩 대상 변수를 제외한 13개(수치형+라벨)에 알고리즘을 적용한 결과 11개의 변수를 선택하였고, 원핫인코딩 대상 변수와 결합하여 최종 데이터 구성

i Isolation Forest 검증(T-SNE 적용) 및 최종 데이터 구성

수치형 변수에 **T-SNE**를 적용하여 3차원으로 축소한 결과, 이상치 제거(Isolation Forest)가 적절히 작동하였으며, **Boruta** 알고리즘으로 변수 선택까지 마친 후 최종 데이터는 7개의 문자형 변수(원핫인코딩 6 + 라벨인코딩 1) 및 9개의 수치형 변수, 1개의 목적변수(이진분류)로 구성되어 있습니다.



4. 대출 연체여부 예측 모델 구축 및 평가

ADASYN을 이용한 오버샘플링

모델링에 앞서, 클래스 불균형 문제는 ADASYN을 통한 오버샘플링으로 해결하였습니다. 과대평가, 과적합을 방지하고 검증 무결성을 위해 CV 과정의 훈련 데이터에만 오버샘플링하였으며, (imblearn.pipeline 활용) 이를 통해 검증은 항상 원본데이터로만 진행됩니다.

XGBoost 모델 구축

앞서 구성한 40개 변수로 “대출 연체 여부”를 예측하는 모델을 XGBoost 알고리즘을 통해 구축하였으며, 모델 튜닝은 2단계 최적화 접근법을 적용하였습니다. 이러한 방식은 과적합을 방지하고 정해진 계산자원 하에서 최대한 공정하게 파라미터를 비교할 수 있는 장점이 있습니다.

1단계: 초기 하이퍼파라미터 탐색 - 상대적으로 높은 학습률(0.1)과 고정된 n_estimators 값으로 하이퍼파라미터 조합을 탐색 - 각 조합이 동일한 학습 기회(같은 트리 개수)를 갖도록 보장 (CV 내에서 ADASYN 오버샘플링 적용)

2단계: 최적 모델 미세 조정 - 1단계에서 찾은 최적 하이퍼파라미터에 낮은 학습률(0.01)과 높은 n_estimators(10000) 적용 - 조기종료를 적용(50)하여 최적의 트리 개수 결정하고, 전체 훈련/검증 데이터를 사용하여 모델 학습

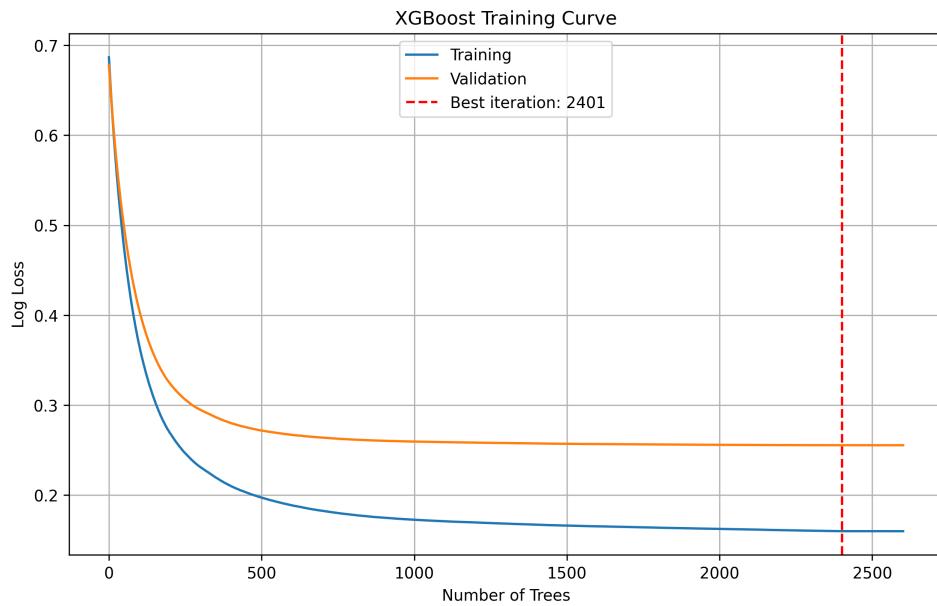


Figure 2.1: 2단계 접근법의 최종 모델 학습곡선

모델 일반화성능 평가

클래스 불균형 문제를 고려하여 최종 모델의 일반화 성능을 F1-Score를 중심으로 평가하도록 하겠습니다. F1-score는 약 0.94, ROC-AUC는 약 0.91로 실제 연체 여부를 잘 예측하는 것으로 나타났습니다.

특히, 모델 튜닝 과정에서 1단계 하이퍼파라미터 탐색시 최고 F1-Score가 0.93544였는데, 일반화 성능은 그와 동일한 수준이므로 과적합 방지를 위한 2단계 최적화 기법이 효과적인 것을 알 수 있습니다.

<< 최종 XGBoost 모델 성능 >>
F1 Score: 0.9354695016927006
ROC AUC: 0.9081721636844388

분류 보고서:

	precision	recall	f1-score	support
0	0.92	0.48	0.63	23302
1	0.89	0.99	0.94	95507
accuracy			0.89	118809
macro avg	0.90	0.74	0.78	118809
weighted avg	0.89	0.89	0.88	118809

i 여러 Gradient boosting 계열의 알고리즘과 비교

주로 사용한 XGBoost 이외에도 다양한 Gradient boosting 계열 및 양상을 알고리즘이 존재합니다.

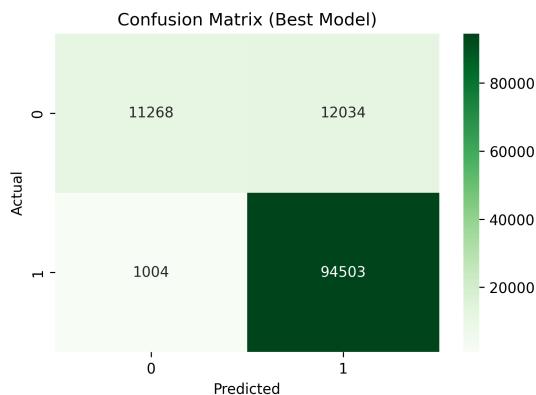
- **XGBoost**: Regularization과 트리 구조 최적화에 강점을 가진 Gradient Boosting 모델 |
- **CatBoost**: 범주형 변수 자동 인식 기능이 있는 Gradient Boosting 기반 모델
- **LightGBM**: 빠른 학습 속도와 낮은 메모리 사용의 Gradient Boosting 기반 모델
- **Soft Voting**: CatBoost, LightGBM의 예측 확률 평균을 통한 결합(양상을) 모델
- **Stacking**: CatBoost, LightGBM의 예측 결과를 Logistic Regression에 전달하는 메타 모델 기반 양상을

XGBoost와 유사한 방식으로 각 모델을 튜닝, 훈련하였으며 일반화성능은 유사한 수준이었습니다. F1-Score는 양상을(Soft Voting) 모델이, ROC-AUC 점수는 XGBoost가 가장 우수하였습니다.

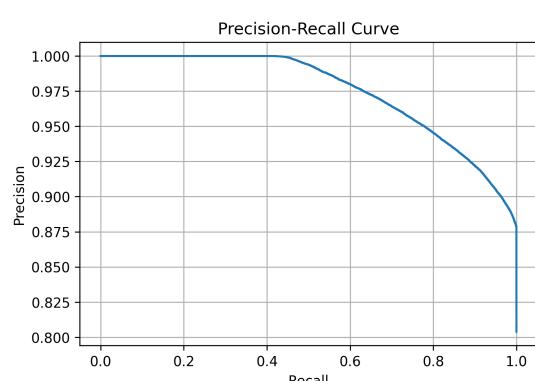
모델	F1 Score	ROC AUC	모델	F1 Score	ROC AUC
CatBoost	0.9355	0.9073	Soft Voting	0.9356	0.9072
LightGBM	0.9353	0.9066	Stacking	0.9330	0.9072

그러나, 샘플이 적은 “부도”인 경우, 예측 성능이 다소 떨어지는 모습이 관측되었습니다.

부도의 절반 이상이 정상으로 분류되었으며 모든 모델에 동일한 문제가 있는 것으로 볼 때, 데이터의 한계인 것으로 보입니다. 또는 신경망 계열을 적용해보는 것도 개선방법이 될 수 있습니다.



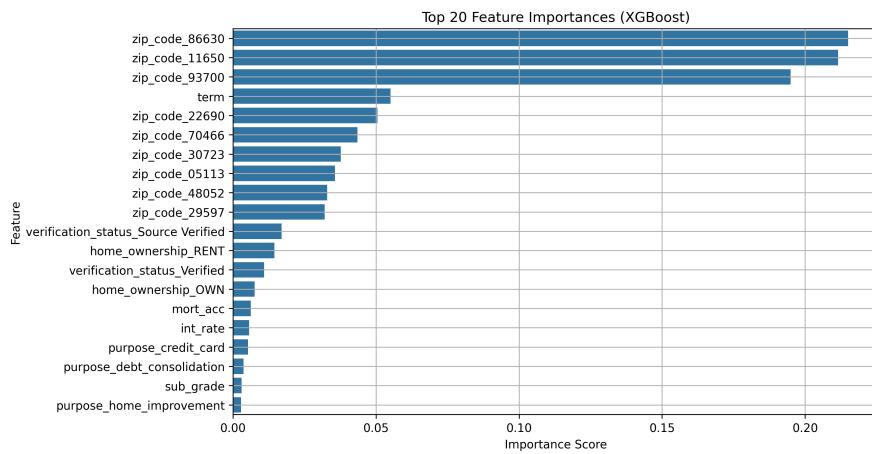
(a) XGBoost Confusion Matrix



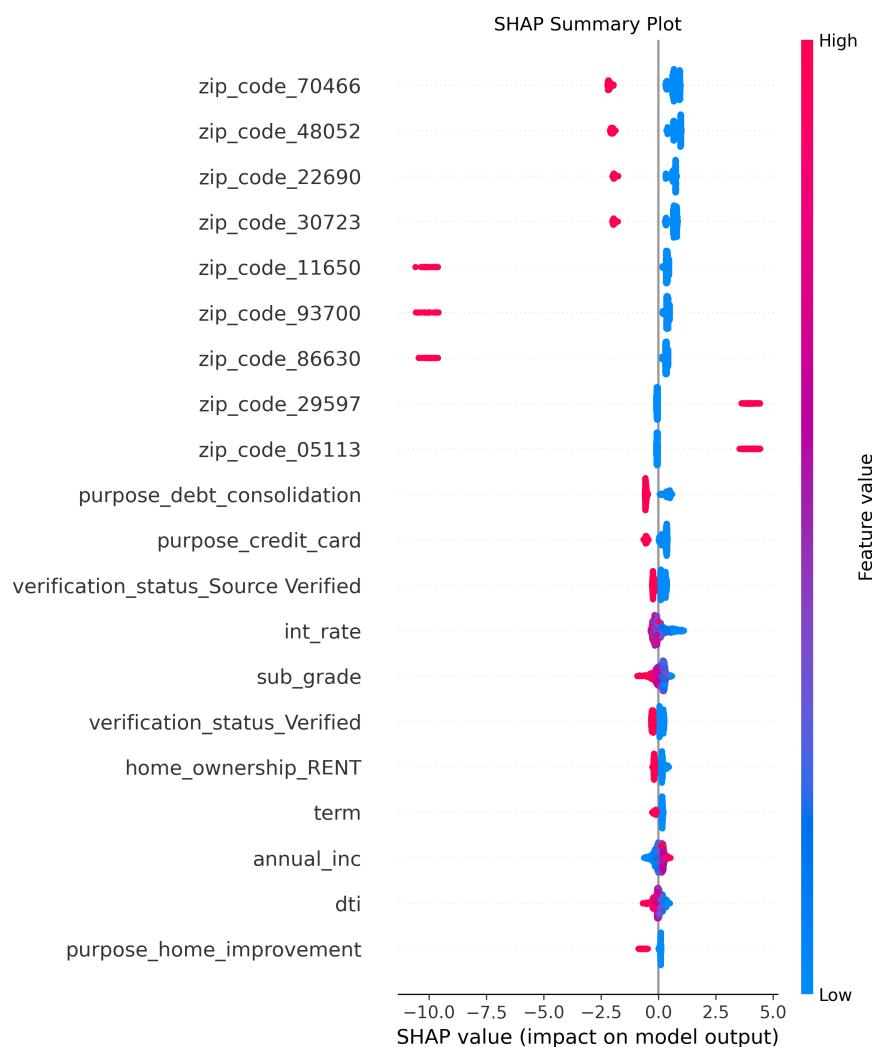
(a) XGBoost PRCurve

변수 중요도(Feature Importance) 분석

기본 변수 중요도 산출 결과 부도 여부에는 예상 외로 주거지가 큰 영향을 미치는 것을 확인할 수 있었으며, 이외에도 대출기간 등이 분류에 영향을 미치는 것을 알 수 있었습니다.



보다 상세한 변수 중요도 분석을 위해 SHAP(SHapley Additive exPlanations) 분석을 실시하여 각 변수의 중요도, 예측에 미치는 영향(방향성, 정도)을 분석하였습니다.



전체적인 중요도는 주거지가 큰 영향을 미친다는 점에서 유사하였으며, 변수별 특징은 아래와 같습니다.

- 상위 7개는 해당 지역에 주거(1) 시, 연체(0)가 많으므로 집값/소득수준이 낮은 주거지로 추정
- 하위 2개는 집값/소득수준이 높은 주거지로 추정
- 대출목적이 부채상환/신용카드, 대출이자가 높고, 대출기간이 길고, 소득이 낮을수록 연체 예측 가능성 증가
- 이 외에도 대부분의 변수가 대출과 관련된 일반적인 직관과 동일한 것을 알 수 있었음

시사점

이번 프로젝트를 통해 실제 은행의 데이터를 살펴보고, 대출의 부도 확률 예측 모델을 구축해보았습니다.

먼저 실제 데이터를 전처리하는 과정에서 발생하는 결측치, 이상치, 적합하지 않은 변수 분류 등의 문제점을 실제로 경험할 수 있었고 Isolation Forest 및 T-SNE, Boruta 알고리즘을 적용해보면서 각 알고리즘이 어떻게 작동하는지, 어떤 방식으로 문제를 해결하고 활용되는지 알 수 있었습니다.

또한, XGBoost 알고리즘이 금융데이터 예측에 강력한 성능을 가진 것을 확인하였고, 모델 성능에는 알고리즘 선택 뿐만아니라 하이퍼파라미터 튜닝 방법(2단계 최적화) 클래스 불균형 해소(oversampling), 변수 선별(boruta) 등이 매우 중요하다는 것을 느꼈습니다.

결과적으로 은행 대출의 연체여부에는 주거지가 매우 큰 영향을 미친것으로 나타났으며, 이는 주거지에 집값/주거형태/소득/신용점수/직업 등 종합적인 요소가 모두 반영되어있기 때문인 것으로 추정됩니다. 이외에도 이자율, 대출목적/기간 등이 연체 여부 예측에 영향을 미치는 것으로 나타났으며, 그 영향은 일반적인 직관과 동일하였습니다.

데이터 자체의 한계점으로 인해 소수 표본에 대한 학습이 부족하여 예측력이 다소 떨어지는 한계점이 있었으나, 전반적으로 수업시간에 다룬 여러 알고리즘을 통해 이론이 실제 세상에 적용되는 과정을 이해할 수 있었습니다. 또한, 분석에 적합한 데이터를 구하고 전처리하는 것이 매우 중요하다는 것을 알게 된 프로젝트였습니다.

Part II

알고리즘 거래전략('25 봄)

알고리즘 거래전략과 고빈도 금융

Part III

사례로 보는 금융공학('25 봄)

사례로 보는 금융공학 실무

코스피200 변동성 조정 위클리 양매도 전략

25년도 봄학기 금융공학 특수논제 A조

김경재(20259048) / 강상묵(20259013) / 김형환(20249132) / 어환석(20259248) / 유수형(20259273)

1. 개요

본 리포트는 옵션과 변동성지수를 활용한 변동성 조정 위클리 양매도전략을 알아보고, 과거 데이터를 통해 구현 및 검증하기 위해 작성되었습니다. 변동성 조정 위클리 양매도란 1.변동성 조정, 2.주단위 매도 두가지 특징을 통해 기존 단점을 보완한 양매도 전략으로, “매주” V-Kospi200를 이용해 행사가격 범위를 결정하고 콜/풋옵션을 매도(양매도)하게 됩니다.

전략 소개에 앞서 필요한 배경지식을 다루고, 전략 소개 및 구현, 백테스팅 및 성과를 차례로 설명하겠습니다.

2. 배경지식

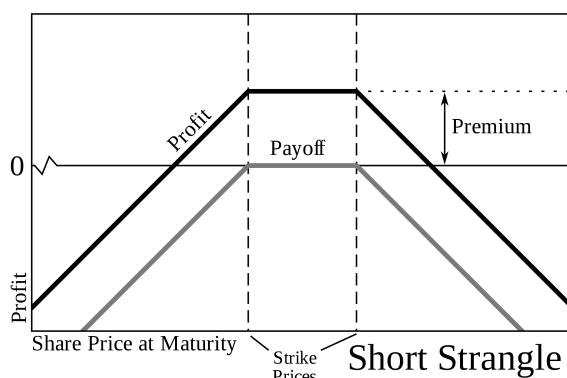
양매도 (Short strangle)

양매도란 옵션 거래전략으로, 일반적으로 만기일이 같은 외가격(OTM) 콜/풋옵션을 매도하는 것을 의미합니다.

OTM 옵션을 매도하므로, 만기일에 기초자산의 가격이 풋옵션의 행사가격과 콜옵션의 행사가격 사이라면 권리행사되지 않게 되고 옵션 프리미엄(매도수익)을 그대로 얻을 수 있게 됩니다.

반면, 양매도는 시장의 변동성이 예상과 달리 큰 경우, 큰 손실이 발생할 수 있다는 단점도 존재합니다. 수익은 프리미엄으로 한정되어있으나, 시황 급변에 따른 옵션 손실에는 제한이 없어 원금 이상의 손실이 발생할 수도 있기 때문입니다.

즉, 양매도는 높은 확률로 안정적인 중수익을 보장하나 매우 낮은 확률로 큰 손실이 발생할 수 있는 전략이며, 향후 시장의 변동성이 크지 않을 것이라 예측될 때 주로 사용됩니다.



양매도 전략 예시

국내의 경우 행사가격의 상하단을 등가격(ATM) ± 5%로 설정한 양매도 ETN이 한 때 큰 인기를 끌었습니다. 대표적인 예시인 월별 코스피 200 OTM 5% 양매도 전략의 거래방법을 살펴보면, 다음과 같습니다.

1. 현재 코스피 200 지수를 기준으로 ATM+5%의콜옵션과 ATM-5%의풋옵션을 매도 (프리미엄 수익 발생)
2. 다음달 만기일이 되면, 옵션은 청산(권리행사시 손실)되고 다시 ATM ± 5%의 콜, 풋옵션을 매도

즉, 월단위로 옵션을 교체하게 되며 행사가격의 상하단은 ATM ± 5%로 고정한 양매도 전략입니다. 중위험-중수익으로 흥행에 성공한 상품이지만, 아래와 같은 한계점도 존재합니다.

1. 행사가격 범위가 고정되어 있어 변동성 확대시 손실 가능성이 커지고, 변동성 축소시 프리미엄 수익이 크게 감소
2. 옵션 교체주기가 1개월로, 그간 지수의 하락이 누적된다면 손실이 과도하게 커질 수 있음 (유동성 리스크)

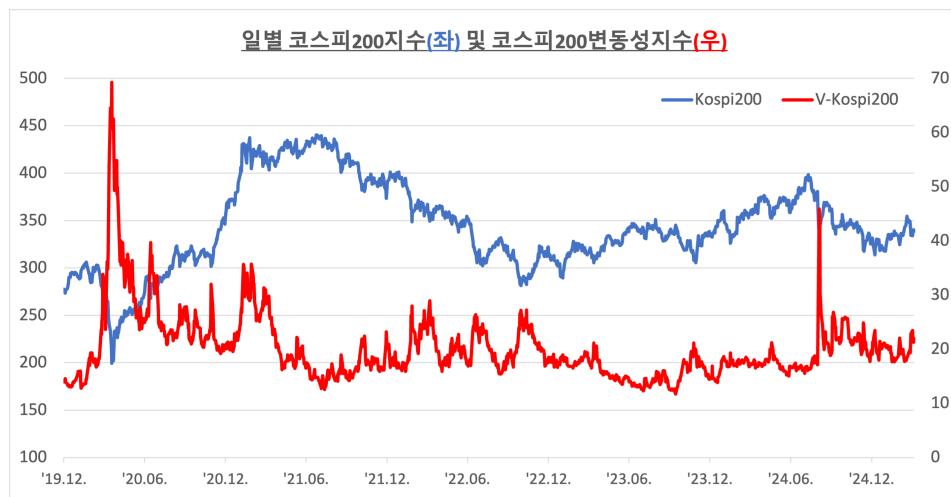
코스피 200 변동성지수 (V-Kospi200 index)

코스피 200 변동성 지수란 주식 시장의 변동성을 측정하는 지표입니다. 코스피 200 옵션 가격 기반의 내재변동성을 이용하여 산출되며, 향후 30일간 시장 변동성에 대한 투자자들의 기대를 나타내는 지수입니다.

주요 특징

- 시장 심리 반영: 투자자들의 불안 심리를 반영하며, '공포 지수(Fear Index)'라고도 불림
- 옵션 가격 기반: 코스피 200 옵션 가격 기반의 내재변동성을 통해 산출되며, 여기에는 투자자들의 기대가 반영

그래프 1 : 일별 코스피 200 지수 및 코스피 200 변동성지수



코스피 200 위클리옵션

코스피 200 위클리옵션이란 코스피 200 지수를 기초자산으로 하는 주간 단위의 옵션 거래 상품을 말합니다. '19년 상장되었으며 기존의 월 단위 만기가 도래하는 옵션과는 달리, 매주 월/목요일에 만기가 도래하는 단기 옵션입니다.

주요 특징

- 세밀한 거래 지원: 매주 월/목요일에 만기가 도래하여 단기적인 시장 변동에 대한 투자 및 위험 관리에 유리
- 다양한 투자 전략 활용: 단기적인 시장 예측을 기반으로 다양한 투자 전략을 구사할 수 있습니다.

3. 전략 소개 및 구현

전략 개요

변동성 조정 위클리 양매도(Volatility Adjusted Weekly Short strangle, VWss) 전략이란, 기존에 널리 활용되던 월별 OTM 5% 양매도 전략에 아래 두가지 방법을 결합한 전략을 의미합니다.

1. 옵션의 교체주기를 “매월” -> “매주” 단위로 세분화
2. 행사가격의 범위를 고정하는 것이 아니라 매 옵션 매도시점마다 시장 변동성을 고려하여 조정

따라서, 기존과 달리 주단위로 옵션을 교체하며, 행사가격의 범위는 교체시점에 매번 달라지게 됩니다. 이를 위해 코스피 200위클리옵션을 사용하고, 변동성 지표는 내재변동성 기반의 V-Kospi200지수를 사용**하여 구현할 계획입니다.

행사가격 범위 설정방법

행사가격의 상하단은 옵션 매도시점의 “전일 V-Kospi200 종가”를 참조하여 설정할 예정입니다. V-Kospi200는 향후 30 일간 코스피200 지수의 변동성을 수치화한 지표로서, 현재시점에서 변동성을 잘 예측할 수 있는 수단이기 때문입니다.

다만, 본 전략에서 옵션 매도포지션의 유지기간은 약 7일이므로 지수 종가(연율)를 주단위 기간에 맞도록 환산하여 행사 가격의 상하단을 산출하였습니다.

$$\sigma_{Target} = \frac{VKospi200_{(T-1)} \times \sqrt{Calendar\ days}}{\sqrt{365}}$$

$$Call\ strike = Kospi200_{(T)} \times (1 + \sigma_{Target}) \ rounded\ up\ to\ 2.5pt$$

$$Put\ strike = Kospi200_{(T)} \times (1 - \sigma_{Target}) \ rounded\ down\ to\ 2.5pt$$

이렇게 행사가격을 설정하면 V-Kospi200가 15pt일 때 월환산 변동성이 약 5%(주환산 2.5%)가 됩니다. 즉,

- 시장의 예상 변동성(V-Kospi200)이 연 15% 이상 -> 행사가격 범위를 기준보다 넓게 설정하여 손실 가능성 축소
- 시장의 예상 변동성이 연 15% 미만 -> 행사가격 범위를 기준보다 좁게 설정하여 프리미엄 수익 극대화

본 전략은 위클리옵션을 사용하므로 향후 1주일 변동성이 필요합니다. 따라서, 30일 변동성을 나타내는 V-Kospi200은 만기 mismatch가 있지만, 단기간의 예측치에는 큰 차이가 없고 이외의 대안(e.g. 7-day VIX)이 없어 V-Kospi200를 그대로 사용하였습니다.

포트폴리오 구성 방법

먼저, 편의를 위해 세금/수수료/호가스프레드 등의 거래비용은 없으며 종가에 원하는 수량만큼 거래할 수 있는 완전자본 시장을 가정하도록 하겠습니다. 전략 구현을 위한 포트폴리오(투자원금 100억원) 구성 과정은 아래와 같습니다.

1. 전일 V-Kospi200 지수를 주단위로 환산하여 예상변동성($\hat{\sigma} = (VKospi200 \times \sqrt{day})/\sqrt{365}$) 산출
2. 예상변동성을 당일 Kospi200지수에 적용하여 행사가격 상하단(ATM $\pm \hat{\sigma}$ 을 2.5pt 단위로 올림/내림) 산출
3. 당일 Kospi200지수 및 승수(25만)를 적용, 옵션 매도수량($Q_{sell} = nominal/(kospi200 \times multiplier)$) 산출
4. 행사가격 상단 콜옵션, 하단 풋옵션 매도 ($Premium = (callprice + putprice) \times Q_{sell} \times multiplier$)
5. 원금과 매도수익을 다음주 만기일까지 MMF에 투자 ($Interest = (nominal + Premium) \times MMF \times \frac{day}{365}$)
6. 다음주 만기일이 되면, 권리행사 손실을 포함하여 최종손익 산출($Revenue = Premium + Interest - Exercise$)
 - Kospi200 > 행사가격 상단, 콜옵션에서 손실 발생($Exercise = (Kospi200 - Strike_{call}) \times multiplier$)
 - Kospi200 < 행사가격 하단, 풋옵션에서 손실 발생($Exercise = (Strike_{put} - Kospi200) \times multiplier$)
 - 이외의 경우, 권리행사되지 않아 옵션 포지션 청산($Exercise = 0$)
7. 주간 최종손익을 정산($nominal_{new} = nominal_{old} + Revenue$)하고, 투자종료시점까지 1. ~ 6. 과정을 반복

3. Backtesting

데이터 수집 및 전처리, 포트폴리오 구현

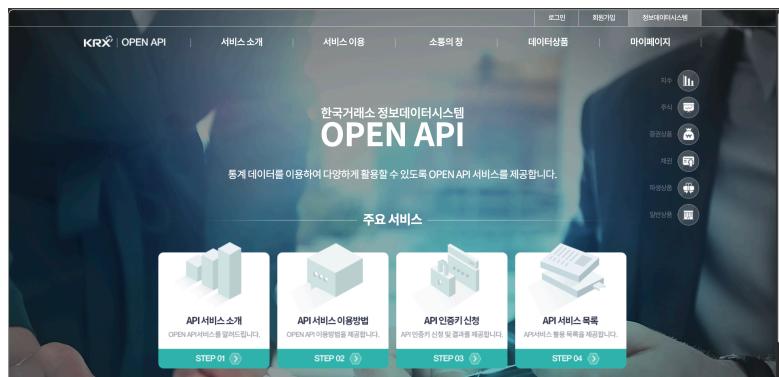
과거 5개년(2020~2024) 코스피200 등의 지수/옵션 가격, 금리를 수집하였으며, 출처는 아래와 같습니다.

한국거래소 정보데이터시스템(data.krx.co.kr) : 코스피200 및 V-Kospi200지수

한국거래소 OpenAPI(openapi.krx.co.kr) : 코스피200옵션 및 위클리옵션 종목별 가격

한국은행 경제통계시스템(ecos.bok.or.kr) : 일별 MMF(7일) 금리

i 한국거래소 OpenAPI를 활용한 데이터 수집 예시



API 호출시 json 데이터를 수집할 수 있으며, 이를 Dataframe(python) 및 tibble(r)로 변환하여 활용하였습니다.

```

import requests; import json
url = 'http://data-dbg.krx.co.kr/svc/sample/apis/drv/opt_bydd_trd?basDd=20250312'
headers = {'AUTH_KEY': '74D1B99DFBF345BBA3FB4476510A4BED4C78D13A'}
res = requests.get(url=url, headers=headers); res.text

'{"OutBlock_1": [{"BAS_DD": "20250312", "PROD_NM": "코스피200 옵션", "RGHT_TP_NM": "CALL", "ISU_CD": "20250312"}]

```

수집한 데이터는 **R**을 이용하여 전처리하였으며, 두개의 데이터셋으로 요약하였습니다.

1. 포트폴리오 기본 정보 : 옵션 만기일(매주), 옵션 보유일, MMF금리, Kospi200, 전일 V-K200, 거래대상옵션 정보

BAS_DD	VOL	DIFF_DAY	MMF	KOSPI200	LAG_VK200	PRIOR	TARGET_C	TARGET_P_K
20241219	0.0248719	7	0.0334	322.38	17.96	24124	332.5	312.5
20241212	0.0285279	7	0.0335	329.04	20.60	24123	340.0	317.5
20241205	0.0295527	7	0.0341	323.87	21.34	24122	335.0	312.5
20241128	0.0252043	7	0.0340	331.45	18.20	24121	340.0	322.5
20241121	0.0276001	7	0.0344	329.49	19.93	24114	340.0	320.0
20241114	0.0344274	7	0.0340	317.70	24.86	24113	330.0	305.0

2. 거래대상옵션 정보 : 포트폴리오 기본 정보에 대응되는 거래대상옵션의 종가, 거래량(0인 경우 제외)

BAS_DD	PRICE_TARGET_C	PRICE_TARGET_P	ACC_TRDVOL_TARGET_C	ACC_TRDVOL_TARGET_P
20241219	0.29	0.74	343	137
20241212	0.42	0.78	190	196
20241205	0.44	0.91	65452	51417
20241128	0.34	0.43	288	201
20241121	0.37	0.53	115	201
20241114	0.43	0.74	329	165

이를 통해 매도수량, 프리미엄, 이자수익, 권리행사손실 등을 산출하여 포트폴리오를 구현하였습니다.

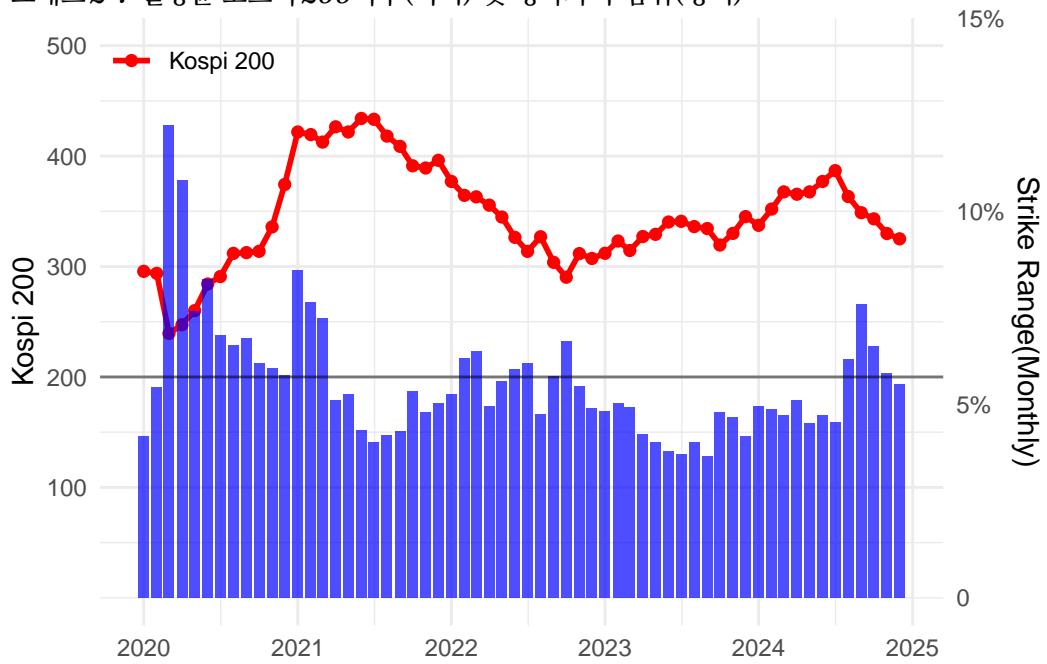
BAS_DD	VOL	SELL_AMTPREMIUM	INTEREST	EXERCISE	REVENUE	RATE
20200102	0.0203434	137.7648	29963837	2789154	0	32752991
20200109	0.0221160	135.8650	20040080	2844044	9510547	13373578
20200116	0.0185154	132.1091	18825550	2824485	0	21650035
20200123	0.0197895	132.3058	23815037	2787444	219296795	-192694314
20200130	0.0235286	138.7107	45080972	2793358	109234664	-61360333
20200206	0.0252458	133.0451	46565774	2774504	0	49340278

! 변동성 조정 방법에 따른 행사가격 범위 추이(과거 5년)

V-Kospi200과 연동한 행사가격의 범위는 지난 5년간 **1.6% ~ 8.7%**까지 광범위하게 형성되었습니다.

코스피200이 급등락하는 경우 범위가 넓게 형성되고 보합장에서는 좁게 형성되는 추이를 확인할 수 있으며, 이를 월환산($\times \sqrt{4}$)하여 기존 5%와 비교하면 지수의 상황에 따라 유동적으로 조절되고 있음을 의미합니다.

그래프 2 : 월평균 코스피 200지수(적색) 및 행사가격 범위(청색)



Backtesting 성과

지난 5년간(2020~2024) VW양매도와 코스피200지수 / OTM 5% 양매도를 비교해보았습니다.

표 1 : 연도별 변동성 조정 위클리 양매도 포트폴리오 및 코스피 200지수 성과

YEAR	Expiry	NoExercise	Premium	Interest	Loss	Revenue	Return	Return_K200
2020	50	0.74	4385	184	5338	-769	-0.04	0.33
2021	47	0.91	2974	142	728	2387	0.12	0.01
2022	52	0.79	3304	421	4152	-427	-0.03	-0.26
2023	52	0.81	2456	732	1758	1430	0.08	0.23
2024	49	0.80	3697	719	3454	962	0.05	-0.11

표 2 : 연도별 일반 양매도 포트폴리오 및 코스피 200지수 성과

YEAR	Expiry	NoExercise	Premium	Interest	Loss	Revenue	Return	Return_K200
2020	12	0.58	16507	847	37851	-20497	-0.23	0.33
2021	12	1.00	9887	605	0	10492	0.13	0.01
2022	12	0.50	10207	1819	15098	-3071	-0.04	-0.26
2023	12	0.83	4591	3176	884	6883	0.09	0.23
2024	12	0.75	9850	3072	9589	3333	0.04	-0.11

YEAR : 산출대상 연도 / **Expiry** : 옵션만기 횟수 (프리미엄 수익 발생 횟수)

NoExercise : 옵션만기일에 행사되지 않은 비율 (손실이 발생하지 않은 거래일 비율)

Premium : 평균 옵션 프리미엄 수익 (만원) / **Interest** : 원금 및 프리미엄에서 발생한 평균 이자수익 (만원)

Loss : 옵션권리행사로 인한 평균 손실 (미행사 포함) / **Revenue** : 평균 이익 (Premium + Interest - Loss)

Return : 포트폴리오의 연환산 수익률 / **Return_K200** : 코스피200지수의 연환산 수익률

변동성 조정 위클리 양매도 포트폴리오의 주요 성과는 다음과 같습니다.

1. 옵션 매도주기 축소(월간 → 주간) : 현금흐름 개선 및 프리미엄 수익 극대화

- VW양매도 포트폴리오는 연평균 50회의 수익이 발생하는 반면, 일반 양매도 포트폴리오는 12회(월1회) 발생.
- 1회 발생 수익은 4배 미만으로 감소하여 평균 수익이 증가하였고, 수익주기가 짧아지면서 현금유동성 개선

2. 행사가격 범위 조정(5% → σ 연동) : 포트폴리오 안정성 증가 및 리스크 축소

- 일반 양매도 전략의 손실발생비율(권리행사비율)은 시장 상황에 따라 0~50%까지 큰 폭으로 변동
- 반면, 본 포트폴리오는 행사가격 범위를 변동성 수준에 조정하므로 비율이 10~30%수준으로 안정화
- 손실에 대한 예측가능성 및 포트폴리오 변동성이 개선되었으며 1회 손실도 감내 가능한 수준으로 축소

3. 소결 : VW양매도 전략은 수익률, 리스크 측면에서 코스피200지수 및 일반 양매도 전략을 Outperform

그래프 3, 4 : 지난 5년 및 3년간 코스피 200지수, VW양매도, 5%양매도 누적수익률 및 초과수익

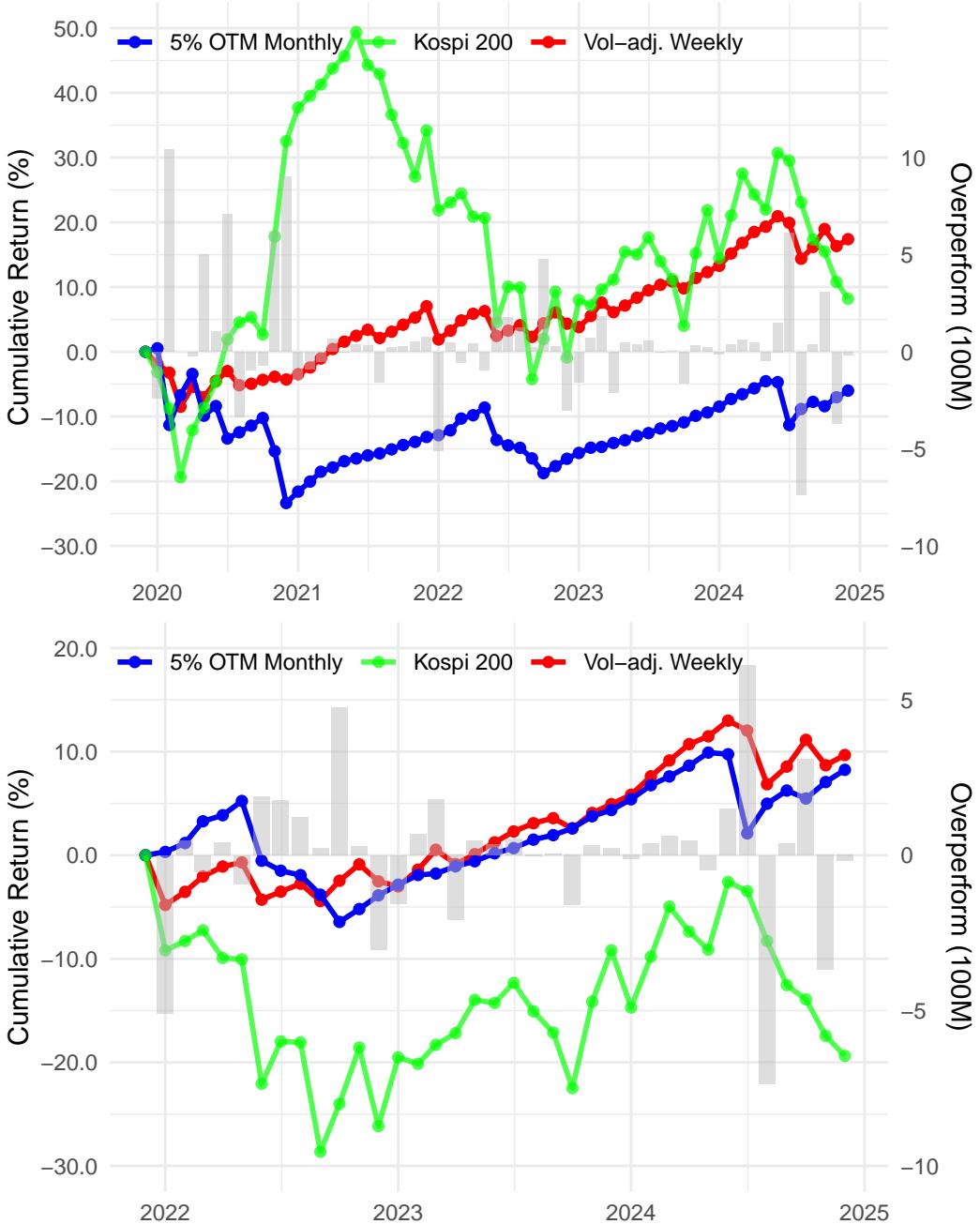


표 3 : 연도별 VW양매도, 일반 양매도, 코스피 200지수의 수익률 및 변동성(월간수익률의 연환산)

YEAR	Return_main	Return_sub	Return_K200	Vol_main	Vol_sub	Vol_K200
2020	-0.04	-0.23	0.33	0.08	0.19	0.27
2021	0.12	0.13	0.01	0.03	0.02	0.11
2022	-0.03	-0.04	-0.26	0.08	0.08	0.25
2023	0.08	0.09	0.23	0.04	0.01	0.17
2024	0.05	0.04	-0.11	0.07	0.08	0.16

그래프와 표를 통해 VW양매도 전략이 타 전략 대비 안정적이고 높은 수익을 실현하였음을 확인할 수 있었습니다.

한계점

그러나, 행사가격 범위가 전일 **V-Kospi200** 지수에 따라 결정되므로 옵션 매도일 및 보유기간 중 V-Kospi200 지수가 급등락하는 경우 시장 변동성을 적절히 반영되지 않을 가능성이 있습니다.

- 당일 장중 지수를 사용할 수 있겠으나 종가보다 신뢰성이 높다고 보기 어렵고, 당일 종가는 매도시점에서 확인 불가
- 보유기간 중 V-Kospi200의 변동에 따라 옵션 행사가격을 리밸런싱하면 보다 정확히 변동성을 반영할 수 있으나, 잊은 거래로 인해 수반되는 비용이 급증
- 시장 변동성이 적절히 반영되지 않는 경우, 프리미엄 수익성이 낮아지고 옵션 권리행사 위험이 증가할 수 있음

또한, 실제로 **VW양매도 포트폴리오**를 운영한다면 거래비용 및 유동성 등의 한계점으로 인해 보완해야 할 점이 있으며 이 과정에서 초과수익률 등의 성과가 다소 희석될 것으로 예상됩니다.

- 수수료/유동성 등으로 인해 자주 옵션을 매도하는 전략 특성상 거래비용 상승이 수반됨
- 월물 옵션 대비 위클리옵션의 유동성이 낮아, 변동성 확대 국면에서 원하는 위클리옵션의 거래가 없을 수 있음

4. 포트폴리오 검증 ('25.3.13 ~ 4.10)

포트폴리오의 성과 검증을 위해 '25년 3월 옵션만기일부터 1개월간의 성과를 측정해보겠습니다.

Backtesting과 동일한 방식으로 진행하였으며, 날짜 등 일부를 제외하고 동일 코드를 재사용하였습니다.

표4 : 검증기간('25.3.13 ~ 4.10) VW양매도 및 일반 양매도 전략 성과

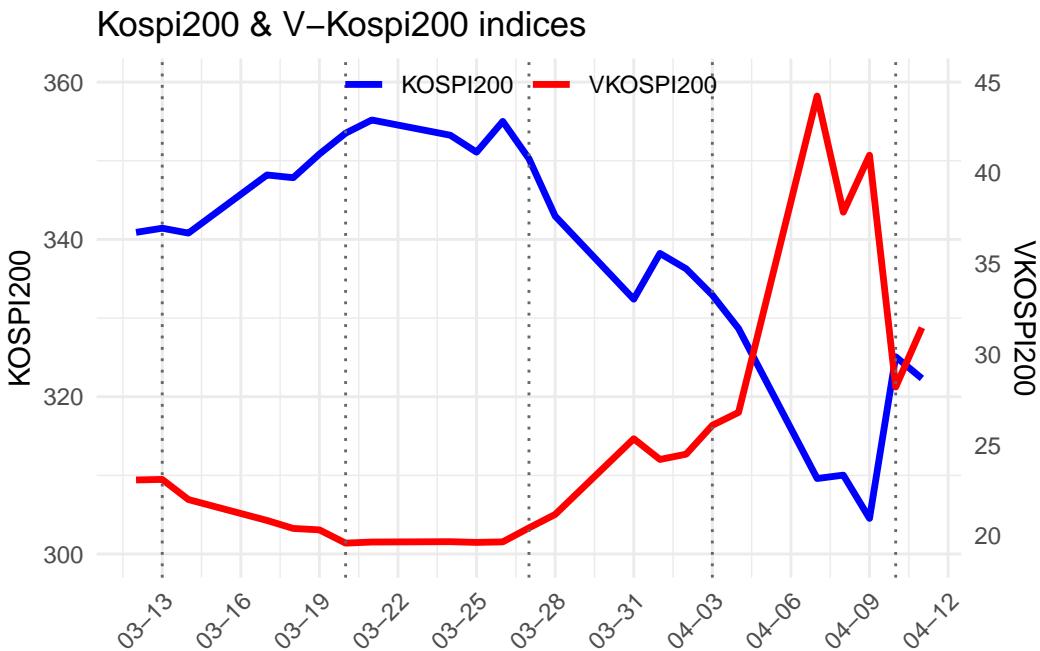
BAS_DD	Group	Premium	Interest	Loss	Revenue	Return
20250313	Monthly	12653	2400	0	15053	0.02
20250313	Vol-adj. Weekly	5624	596	2929	3291	0.00
20250320	Vol-adj. Weekly	2687	587	0	3274	0.00
20250327	Vol-adj. Weekly	4283	578	20214	-15353	-0.02
20250403	Vol-adj. Weekly	11594	578	0	12173	0.01
81001363	Vol-adj. Weekly Sum	24188	2338	23142	3384	0.00

검증기간 중 **VW양매도 전략**은 옵션을 4회 매도하였고, 일반 양매도는 1회 매도하였습니다.

VW양매도 전략의 프리미엄이 4회 발생하면서 수익성은 개선되었지만, 세번째 매도시기에 큰 손실이 발생하면서 순이익이 악화되었습니다. 벡테스팅 결과와 비교할 때 상반된 결과입니다.

그 원인은 최근 트럼프 행정부의 관세 부과 정책의 영향으로, 증시가 이례적으로 급등락한 데에서 찾을 수 있었습니다.

그래프5 : 검증기간('25.3.13 ~ 4.10) Kosp200 및 V-Kosp200 지수 추이



먼저, 3.27일 예상변동성(V-K200)이 낮아 행사가격 범위가 좁게 형성되어 VW양매도 전략이 실행되었고, 이후 관세 정책 영향으로 지수가 급락하면서 큰 손실이 발생하게 되었습니다.

반면 월별 전략은 만기 직전에 관세가 연기되면서 중시가 회복하였고($304 > 325$), 손실을 피하게 되었습니다.

다소 아쉬운 결과이나, VW양매도 전략의 한계점과 특수한 상황에서는 오히려 불리하다는 점을 알 수 있었습니다.

1. VW양매도 전략은 시장 변동성이 적절히 반영되지 않을 가능성이 있고, 이 경우 예기치 못한 손실이 발생할 수 있음
2. 옵션 만기가 짧은 것은 일반적으로 수익성, 유동성 등에서 장점이 있으나, 중시가 급락 후 회복하는 상황에서는 만기가 긴 옵션이 유리할 수 있음

5. Appendix : Python, R code

Python code : 데이터 수집 및 전처리

```
import requests
import json
import pandas as pd
import aiohttp
import asyncio

# KRX OpenAPI 예시
url = 'http://data-dbg.krx.co.kr/svc/sample/apis/drv/opt_bydd_trd?basDd=20250312'
headers = {'AUTH_KEY': '74D1B99DFBF345BBA3FB4476510A4BED4C78D13A'}
res = requests.get(url=url, headers=headers)
res.text

# KRX OpenAPI를 이용한 옵션 데이터 수집 (네트워크 병렬 처리)
idx_data = pd.read_csv('data/idx_data.csv')
url = 'http://data-dbg.krx.co.kr/svc/apis/drv/opt_bydd_trd?basDd='
key = 'BDFA640BBCE84C4B8A465EA024D50D6F3FD909FF'

async def fetch(session, url, bas_dd):
    async with session.get(url + bas_dd, headers={'AUTH_KEY': key}) as response:
        return await response.json()

async def fetch_all(bas_dd_list, url):
    async with aiohttp.ClientSession() as session:
        tasks = [fetch(session, url, str(bas_dd)) for bas_dd in bas_dd_list]
        return await asyncio.gather(*tasks)

async def main():
    bas_dd_list = idx_data['BAS_DD'].astype(str).tolist()
    responses = await fetch_all(bas_dd_list, url)

    data_list = [pd.json_normalize(res['OutBlock_1']) for res in responses if 'OutBlock_1' in
    options = pd.concat(data_list, axis=0, ignore_index=True)

    # CSV 저장
    options.to_csv("options_data.csv", encoding="utf-8-sig", index=False)
```

```
print("complete")
```

R code 1 : 데이터 가공, Backtesting

```
rm(list=ls())
library(tidyverse)
library(knitr)
library(patchwork)

# 1. 원본데이터 준비
options_raw <- read_csv("data/options_data.csv") %>% tibble()
idx_raw <- read_csv("data/idx_data.csv") %>% tibble()
mmf_raw <- read_csv("data/mmf.csv") %>% tibble()

# 2-1. 데이터 전처리 : KRX openAPI 옵션데이터 전처리
options <- options_raw %>%
  # K200, WK200 이외의 옵션 제외
  filter(substr(PROD_NM,1,3)=="코스피") %>%
  # 종가가 없는 경우 익일 기준가격(이론가격) 사용
  mutate(PRICE = as.double(if_else(TDD_CLSPRC=="-",NXTDD_BAS_PRC,TDD_CLSPRC))) %>%
  drop_na(PRICE) %>%
  select(BAS_DD,ISU_NM,PRICE,ACC_TRDVOL) %>%
  separate(ISU_NM, into=c("PROD","RGHT","EXP","EXER_PRC"), sep=' ') %>%
  mutate(EXER_PRC=as.double(EXER_PRC),
        EXPMM;if_else(PROD=="코스피200",substr(EXP,3,6),substr(EXP,1,4)),
        EXPWW;if_else(PROD=="코스피200",2,as.integer(substr(EXP,6,6)))) %>%
  filter(EXER_PRC>min(idx_raw$KOSPI200)*0.85,
        EXER_PRC<max(idx_raw$KOSPI200)*1.15,
        PROD!="코스피워클리M") %>% # 월요일 만기 워클리옵션 제외
  mutate(PRIOR=as.integer(EXPM)*10+EXPWW) # 만기가 짧은 순으로 우선순위 부여

# 2-2. 데이터 전처리 : 옵션 만기일 데이터 생성
target_date <- options %>%
  distinct(.,BAS_DD, PRIOR) %>%
  arrange(PRIOR, desc(BAS_DD)) %>%
  group_by(PRIOR) %>%
  slice(1) %>%
  ungroup() %>%
```

```

distinct(BAS_DD) %>%
arrange(desc(BAS_DD)) %>%
filter(BAS_DD<20241231, BAS_DD>20200101)

# 2-3. 데이터 전처리 : KRX 정보데이터시스템 K200 및 V-K200지수 전처리
idx_data <- idx_raw %>%
arrange(BAS_DD) %>%
mutate(LAG_K200=lag(KOSPI200),
LAG_VK200=lag(VKOSPI200))

K200_portfolio=idx_data %>%
left_join(mmf_raw, by="BAS_DD") %>%
mutate(RATE=(KOSPI200-LAG_K200)/LAG_K200,
YEAR = substr(BAS_DD, 1, 4),
YM = ym(substr(BAS_DD, 1, 6))) %>%
group_by(YEAR, YM) %>%
summarise(RATE_K200 = if_else(is.na(prod(1 + RATE)), 0, prod(1 + RATE)),
MMF = mean(MMF)/100,
.groups = "drop") %>%
ungroup()

# 3-1. 포트폴리오 구성 : VW양매도 포트폴리오 기초정보 생성
target_info <- target_date %>%
left_join(options, by="BAS_DD") %>%
distinct(BAS_DD,PRIOR) %>%
arrange(desc(BAS_DD),PRIOR) %>%
group_by(BAS_DD) %>%
# 만기가 도래하는 옵션을 제외하고 우선순위가 높은 옵션 선택
slice(2) %>%
ungroup() %>%
arrange(desc(BAS_DD)) %>%
left_join(idx_data, by="BAS_DD") %>%
left_join(mmf_raw, by="BAS_DD") %>%
# 옵션 보유기간 및 단기금리(MMF) 계산
mutate(DIFF_DAY=as.integer(ymd(lag(BAS_DD))-ymd(BAS_DD)),
MMF=MMF*0.01) %>%
# 옵션 보유기간에 해당하는 시장기대변동성 계산(V-K200활용)
mutate(VOL=LAG_VK200*sqrt(DIFF_DAY)/sqrt(365)/100) %>%
# 변동성에 따른 옵션 행사가격 상단(2.5단위 올림) 및 하단(2.5단위 내림) 계산

```

```

mutate(TARGET_C_K=ceiling(KOSPI200*(1+VOL)*0.4)/0.4,
       TARGET_P_K=floor(KOSPI200*(1-VOL)*0.4)/0.4) %>%
drop_na(DIFF_DAY) %>%
select(BAS_DD,VOL,DIFF_DAY,MMF,KOSPI200,LAG_VK200,PRIOR,TARGET_C_K,TARGET_P_K)

# 3-2. 포트폴리오 구성 : VW양매도 포트폴리오 거래대상 옵션 정보 생성
target_options <- target_info %>%
  select(BAS_DD, PRIOR, TARGET_C_K, TARGET_P_K) %>%
  pivot_longer(cols=c("TARGET_C_K","TARGET_P_K"),names_to = "GRP", values_to = "EXER_PRC") %>%
  mutate(RGHT=substr(GRP,8,8)) %>%
  left_join(options,by=c("BAS_DD","PRIOR","RGHT","EXER_PRC")) %>%
  select(BAS_DD,GRP,PRICE,ACC_TRDVOL) %>%
  pivot_wider(names_from = "GRP", values_from = c("PRICE","ACC_TRDVOL"))

# 원금 100억원 가정
cash = 10000*10000*100

# 4. 포트폴리오 구현 : VW양매도 전략을 구현하여 일자별 손익 계산
portfolio <- target_info %>%
  left_join(target_options,by="BAS_DD") %>%
  arrange(BAS_DD) %>%
  # 원금에 따른 옵션 매도수량 계산
  mutate(SELL_AMT=cash/250000/KOSPI200) %>%
  # 옵션 프리미엄(매도수익) 계산
  mutate(PREMIUM=SELL_AMT*(PRICE_TARGET_C_K+PRICE_TARGET_P_K)*250000) %>%
  # 원금 및 프리미엄의 MMF 이자수익 및 권리행사손실 계산
  mutate(INTEREST=(cash+replace_na(PREMIUM,0))*MMF*DIFF_DAY/365,
         EXERCISE=(if_else(lead(KOSPI200)-TARGET_C_K>0,lead(KOSPI200)-TARGET_C_K,0) +
                    if_else(TARGET_P_K-lead(KOSPI200)>0,TARGET_P_K-lead(KOSPI200),0))*250000*
  # 주단위 포트폴리오 손익 계산(원금 100억 가정, 당일 투자시 다음주 실현손익을 의미)
  mutate(REVENUE=PREMIUM+INTEREST-EXERCISE) %>%
  # 거래대상 옵션이 상장되어있지 않은 경우, MMF수익만 고려
  mutate(REVENUE;if_else(is.na(REVENUE),INTEREST,REVENUE)) %>%
  mutate(RATE=REVENUE/cash) # 주단위 포트폴리오 수익률

# 5. 비교군 포트폴리오 생성 : 월별 5% OTM 양매도 전략
options2 <- options %>%
  filter(PROD=="코스피200")

```

```

# 옵션 만기일(매달) 생성
target_date2 <- options2 %>%
  distinct(., BAS_DD, PRIOR) %>%
  arrange(PRIOR, desc(BAS_DD)) %>%
  group_by(PRIOR) %>%
  slice(1) %>%
  ungroup() %>%
  distinct(BAS_DD) %>%
  arrange(desc(BAS_DD)) %>%
  filter(BAS_DD<20241231, BAS_DD>20200101)

# 일반 양매도 포트폴리오 기본 정보 생성
target_info2 <- target_date2 %>%
  left_join(options2, by="BAS_DD") %>%
  distinct(BAS_DD, PRIOR) %>%
  arrange(desc(BAS_DD), PRIOR) %>%
  group_by(BAS_DD) %>%
  # 만기가 도래하는 옵션을 제외하고 우선순위가 높은 옵션 선택
  slice(2) %>%
  ungroup() %>%
  arrange(desc(BAS_DD)) %>%
  left_join(idx_data, by="BAS_DD") %>%
  left_join(mmf_raw, by="BAS_DD") %>%
  # 옵션 보유기간 및 단기금리(MMF) 계산
  mutate(DIFF_DAY=as.integer(ymd(lag(BAS_DD))-ymd(BAS_DD)),
         MMF=MMF*0.01,
         VOL=0.05) %>%
  # 변동성에 따른 옵션 행사가격 상단(2.5단위 올림) 및 하단(2.5단위 내림) 계산
  mutate(TARGET_C_K=ceiling(KOSPI200*(1+VOL)*0.4)/0.4,
         TARGET_P_K=floor(KOSPI200*(1-VOL)*0.4)/0.4) %>%
  mutate(DIFF_DAY=if_else(is.na(DIFF_DAY), 28, DIFF_DAY)) %>%
  select(BAS_DD, VOL, DIFF_DAY, MMF, KOSPI200, LAG_VK200, PRIOR, TARGET_C_K, TARGET_P_K)

# 일반 양매도 포트폴리오 거래대상 옵션
target_options2 <- target_info2 %>%
  select(BAS_DD, PRIOR, TARGET_C_K, TARGET_P_K) %>%
  pivot_longer(cols=c("TARGET_C_K", "TARGET_P_K"), names_to = "GRP", values_to = "EXER_PRC") %>%
  mutate(RGHT=substr(GRP, 8, 8)) %>%
  left_join(options2, by=c("BAS_DD", "PRIOR", "RGHT", "EXER_PRC")) %>%

```

```

select(BAS_DD,GRP,PRICE,ACC_TRDVOL) %>%
pivot_wider(names_from = "GRP", values_from = c("PRICE","ACC_TRDVOL"))

# 일반 양매도 포트폴리오 구현
portfolio2 <- target_info2 %>%
left_join(target_options2,by="BAS_DD") %>%
arrange(BAS_DD) %>%
# 원금에 따른 옵션 매도수량 계산
mutate(SELL_AMT=cash/250000/KOSPI200) %>%
# 옵션 프리미엄(매도수익) 계산
mutate(PREMIUM=SELL_AMT*(PRICE_TARGET_C_K+PRICE_TARGET_P_K)*250000) %>%
# 원금 및 프리미엄의 MMF 이자수익 및 권리행사손실 계산
mutate(INTEREST=(cash+replace_na(PREMIUM,0))*MMF*DIFF_DAY/365,
EXERCISE=(if_else(lead(KOSPI200)-TARGET_C_K>0,lead(KOSPI200)-TARGET_C_K,0)+
if_else(TARGET_P_K-lead(KOSPI200)>0,TARGET_P_K-lead(KOSPI200),0))*250000)
mutate(EXERCISE;if_else(is.na(EXERCISE),0,EXERCISE)) %>%
# 주단위 포트폴리오 손익 계산(원금 100억 가정, 당일 투자시 다음주 실현손익을 의미)
mutate(REVENUE=PREMIUM+INTEREST-EXERCISE) %>%
# 거래대상 옵션이 상장되어있지 않은 경우, MMF수익만 고려
mutate(REVENUE;if_else(is.na(REVENUE),INTEREST,REVENUE)) %>%
mutate(RATE=REVENUE/cash) # 주단위 포트폴리오 수익률

# 6. 성과분석 : VW양매도 포트폴리오
performance <- portfolio %>%
drop_na(PREMIUM, EXERCISE) %>%
mutate(YEAR = substr(BAS_DD, 1, 4),
YM = ym(substr(BAS_DD, 1, 6))) %>%
group_by(YEAR, YM) %>%
summarise(PREMIUM = sum(PREMIUM),
INTEREST = sum(INTEREST),
EXERCISE = sum(EXERCISE),
REVENUE = sum(REVENUE),
RATE = prod(1 + RATE),
.groups = "drop") %>%
ungroup()

# 6. 성과분석 : 일반 양매도 포트폴리오
performance2 <- portfolio2 %>%
drop_na(PREMIUM, EXERCISE) %>%

```

```

mutate(YEAR = substr(BAS_DD, 1, 4),
       YM = ym(substr(BAS_DD, 1, 6))) %>%
group_by(YEAR, YM) %>%
summarise(PREMIUM = sum(PREMIUM),
           INTEREST = sum(INTEREST),
           EXERCISE = sum(EXERCISE),
           REVENUE = sum(REVENUE),
           RATE = prod(1 + RATE),
           .groups = "drop") %>%
ungroup()

# 시각화 등
graph1_1 <- performance %>%
arrange(YM) %>%
left_join(K200_portfolio, by=c("YEAR", "YM")) %>%
mutate(CUM_RATE = cumprod(RATE) * 100 - 100,
       CUM_RATE_K200 = cumprod(RATE_K200) * 100 - 100) %>%
bind_rows(tibble(YM=ym(201912), CUM_RATE=0, CUM_RATE_K200=0))

graph1_2 <- performance2 %>%
arrange(YM) %>%
left_join(K200_portfolio, by=c("YEAR", "YM")) %>%
mutate(CUM_RATE = cumprod(RATE) * 100 - 100,
       CUM_RATE_K200 = cumprod(RATE_K200) * 100 - 100) %>%
bind_rows(tibble(YM=ym(201912), CUM_RATE=0, CUM_RATE_K200=0))

graph1_3 <- graph1_1 %>%
inner_join(graph1_2, by = "YM", suffix = c("_1", "_2")) %>%
mutate(REVENUE_DIFF = (REVENUE_1 - REVENUE_2) / 1e8)

graph2_1 <- performance %>%
arrange(YM) %>%
left_join(K200_portfolio, by=c("YEAR", "YM")) %>%
filter(as.integer(YEAR) > 2021) %>%
mutate(CUM_RATE = cumprod(RATE) * 100 - 100,
       CUM_RATE_K200 = cumprod(RATE_K200) * 100 - 100) %>%
bind_rows(tibble(YM=ym(202112), CUM_RATE=0, CUM_RATE_K200=0))

```

```

graph2_2 <- performance2 %>%
  arrange(YM) %>%
  left_join(K200_portfolio, by=c("YEAR", "YM")) %>%
  filter(as.integer(YEAR) > 2021) %>%
  mutate(CUM_RATE = cumprod(RATE) * 100 - 100,
        CUM_RATE_K200 = cumprod(RATE_K200) * 100 - 100) %>%
  bind_rows(tibble(YM=ym(202112), CUM_RATE=0, CUM_RATE_K200=0))

graph2_3 <- graph2_1 %>%
  inner_join(graph2_2, by = "YM", suffix = c("_1", "_2")) %>%
  mutate(REVENUE_DIFF = (REVENUE_1 - REVENUE_2) / 1e8)

graph3_1 <- portfolio %>%
  mutate(YM = ym(substr(BAS_DD, 1, 6))) %>%
  group_by(YM) %>%
  summarise(TargetVol=mean(VOL),
            Kospi200=mean(KOSPI200)) %>% ungroup()

graph1=ggplot() +
  geom_line(data = graph1_1, aes(x = YM, y = CUM_RATE, color = "Vol-adj. Weekly"), size = 1) +
  geom_point(data = graph1_1, aes(x = YM, y = CUM_RATE, color = "Vol-adj. Weekly"), shape = 16) +
  geom_line(data = graph1_2, aes(x = YM, y = CUM_RATE, color = "5% OTM Monthly"), size = 1) +
  geom_point(data = graph1_2, aes(x = YM, y = CUM_RATE, color = "5% OTM Monthly"), shape = 16) +
  geom_line(data = graph1_1, aes(x = YM, y = CUM_RATE_K200, color = "Kospi 200"), size = 1, alpha = 0.5) +
  geom_point(data = graph1_1, aes(x = YM, y = CUM_RATE_K200, color = "Kospi 200"), shape = 16) +
  geom_bar(data = graph1_3, aes(x = YM, y = REVENUE_DIFF * 3),
           stat = "identity", alpha = 0.5, fill = "gray") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  scale_y_continuous(limits = c(-30, 50), breaks = seq(-30, 50, 10), labels = scales::number_format(),
                     sec.axis = sec_axis(~ . / 3, name = "Overperform (100M)", breaks = c(-10, 10, 20, 30, 40, 50))) +
  scale_color_manual(values = c("Vol-adj. Weekly" = "red", "5% OTM Monthly" = "blue", "Kospi 200" = "gray")) +
  labs(x = NULL, y = "Cumulative Return (%)", color = "") +
  theme_minimal() +
  theme(legend.position = c(0.4, 0.93), legend.direction = "horizontal")

graph2=ggplot() +
  geom_line(data = graph2_1, aes(x = YM, y = CUM_RATE, color = "Vol-adj. Weekly"), size = 1) +
  geom_point(data = graph2_1, aes(x = YM, y = CUM_RATE, color = "Vol-adj. Weekly"), shape = 16) +
  geom_line(data = graph2_2, aes(x = YM, y = CUM_RATE, color = "5% OTM Monthly"), size = 1) +

```

```

geom_point(data = graph2_2, aes(x = YM, y = CUM_RATE, color = "5% OTM Monthly"), shape = 16,
geom_line(data = graph2_1, aes(x = YM, y = CUM_RATE_K200, color = "Kospi 200"), size = 1, al
geom_point(data = graph2_1, aes(x = YM, y = CUM_RATE_K200, color = "Kospi 200"), shape = 16,
geom_bar(data = graph2_3, aes(x = YM, y = REVENUE_DIFF * 3),
          stat = "identity", alpha = 0.5, fill = "gray") +
scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
scale_y_continuous(limits = c(-30, 20), breaks = seq(-30, 30, 10), labels = scales::number_f
                  sec.axis = sec_axis(~ . / 3, name = "Overperform (100M)", breaks = c(-10,
scale_color_manual(values = c("Vol-adj. Weekly" = "red", "5% OTM Monthly" = "blue", "Kospi 2
labs(x = NULL, y = "Cumulative Return (%)", color = "") +
theme_minimal() +
theme(legend.position = c(0.4, 0.93), legend.direction = "horizontal")

graph3=ggplot() +
  geom_line(data = graph3_1, aes(x = YM, y = Kospi200, color = "Kospi 200"), size = 1) +
  geom_point(data = graph3_1, aes(x = YM, y = Kospi200, color = "Kospi 200"), shape = 16, size
  geom_bar(data = graph3_1, aes(x = YM, y = TargetVol * 7000),
            stat = "identity", alpha = 0.7, fill = "blue") +
  geom_hline(yintercept = 200, size=0.5, color = "black", alpha=0.5) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  scale_y_continuous(limits = c(0, 500), breaks = seq(100, 500, 100),
                     sec.axis = sec_axis(~ . / 7000, name = "Strike Range(Monthly)", breaks =
  scale_color_manual(values = c("Kospi 200" = "red")) +
  labs(x = NULL, y = "Kospi 200", color = "") +
  theme_minimal() +
  theme(legend.position = c(0.1, 0.93), legend.direction = "horizontal")

```

요약표 1 : VW포트폴리오

```

table1 <- portfolio %>%
  drop_na(PREMIUM, EXERCISE) %>%
  mutate(YEAR = substr(BAS_DD, 1, 4),
         CNT=1,
         NO;if_else(EXERCISE==0,1,0)) %>%
  group_by(YEAR) %>%
  summarise(Expiry = sum(CNT),
            NoExercise = round(sum(NO)/sum(CNT),2),
            Premium = round(mean(PREMIUM)/10000,0),
            Interest = round(mean(INTEREST)/10000,0),
            Loss = round(mean(EXERCISE)/10000,0),

```

```

Revenue = round(mean(REVENUE)/10000,0),
Return = round(prod(1 + RATE)-1,2),
.groups = "drop") %>%
left_join(K200_portfolio %>% group_by(YEAR) %>% summarise(Return_K200=round(prod(RATE_K200)-
by="YEAR"))

# 요약표 2 : 일반 포트폴리오
table2 <- portfolio2 %>%
  drop_na(PREMIUM, EXERCISE) %>%
  mutate(YEAR = substr(BAS_DD, 1, 4),
        CNT=1,
        NO;if_else(EXERCISE==0,1,0)) %>%
  group_by(YEAR) %>%
  summarise(Expiry = sum(CNT),
            NoExercise = round(sum(NO)/sum(CNT),2),
            Premium = round(mean(PREMIUM)/10000,0),
            Interest = round(mean(INTEREST)/10000,0),
            Loss = round(mean(EXERCISE)/10000,0),
            Revenue = round(mean(REVENUE)/10000,0),
            Return = round(prod(1 + RATE)-1,2),
            .groups = "drop") %>%
left_join(K200_portfolio %>% group_by(YEAR) %>% summarise(Return_K200=round(prod(RATE_K200)-
by="YEAR"))

# 요약표 3 : 포트폴리오 변동성 비교
Vol1 <- graph1_1 %>% group_by(YEAR) %>% summarise(Vol=round(sd(RATE)*sqrt(12),2),
                                                       Vol_K200=round(sd(RATE_K200)*sqrt(12),2))
Vol2 <- graph1_2 %>% group_by(YEAR) %>% summarise(Vol=round(sd(RATE)*sqrt(12),2))

table3 <- table1 %>%
  select(YEAR,Return,Return_K200) %>%
  left_join(Vol1, by="YEAR") %>%
  mutate(Return_main=Return,
         Vol_main=Vol) %>%
  select(YEAR, Return_main,Vol_main,Return_K200,Vol_K200) %>%
  left_join(table2 %>%
              select(YEAR,Return) %>%
              left_join(Vol2, by="YEAR") %>%
              mutate(Return_sub=Return,

```

```

    Vol_sub=Vol) %>%
  select(YEAR,Return_sub,Vol_sub), by="YEAR")

```

R code 2 : 전략 검증

```

rm(list=ls())
library(tidyverse)
library(knitr)
library(patchwork)
setwd("homepage/study_25spring/data/")

# 1. 원본데이터 준비
options_raw <- read_csv("options_data_test.csv") %>% tibble()
idx_raw <- read_csv("idx_data_test.csv") %>% tibble()
mmf_raw <- read_csv("mmf_test.csv") %>% tibble()

# 2-1. 데이터 전처리 : KRX openAPI 옵션데이터 전처리
options <- options_raw %>%
  # K200, WK200 이외의 옵션 제외
  filter(substr(PROD_NM,1,3)=="코스피") %>%
  # 종가가 없는 경우 익일 기준가격(이론가격) 사용
  mutate(PRICE = as.double(if_else(TDD_CLSPRC=="-",NXTDD_BAS_PRC,TDD_CLSPRC))) %>%
  drop_na(PRICE) %>%
  select(BAS_DD,ISU_NM,PRICE,ACC_TRDVOL) %>%
  separate(ISU_NM, into=c("PROD","RGHT","EXP","EXER_PRC"), sep=' ') %>%
  mutate(EXER_PRC=as.double(EXER_PRC),
        EXPMM=if_else(PROD=="코스피200",substr(EXP,3,6),substr(EXP,1,4)),
        EXPWW=if_else(PROD=="코스피200",2,as.integer(substr(EXP,6,6)))) %>%
  filter(EXER_PRC>min(idx_raw$KOSPI200)*0.85,
        EXER_PRC<max(idx_raw$KOSPI200)*1.15,
        PROD!="코스피우클리M") %>% # 월요일 만기 위클리옵션 제외
  mutate(PRIOR=as.integer(EXPM)*10+EXPWW) # 만기가 짧은 순으로 우선순위 부여

# 2-2. 데이터 전처리 : 옵션 만기일 데이터 생성
target_date <- options %>%
  distinct(.,BAS_DD, PRIOR) %>%
  arrange(PRIOR, desc(BAS_DD)) %>%
  group_by(PRIOR) %>%

```

```

slice(1) %>%
ungroup() %>%
distinct(BAS_DD) %>%
arrange(desc(BAS_DD)) %>%
filter(BAS_DD<20250411)

# 2-3. 데이터 전처리 : KRX 정보데이터시스템 K200 및 V-K200지수 전처리
idx_data <- idx_raw %>%
arrange(BAS_DD) %>%
mutate(LAG_K200=lag(KOSPI200),
LAG_VK200=lag(VKOSPI200))

K200_portfolio=idx_data %>%
left_join(mmf_raw, by="BAS_DD") %>%
mutate(RATE=(KOSPI200-LAG_K200)/LAG_K200,
YEAR = substr(BAS_DD, 1, 4),
YM = ym(substr(BAS_DD, 1, 6))) %>%
group_by(YEAR) %>%
filter(is.na(RATE)==FALSE) %>%
summarise(RATE_K200 = prod(1 + RATE),
MMF = mean(MMF)/100,
.groups = "drop") %>%
ungroup()

# 3-1. 포트폴리오 구성 : VW양매도 포트폴리오 기초정보 생성
target_info <- target_date %>%
left_join(options, by="BAS_DD") %>%
distinct(BAS_DD,PRIOR) %>%
arrange(desc(BAS_DD),PRIOR) %>%
group_by(BAS_DD) %>%
# 만기가 도래하는 옵션을 제외하고 우선순위가 높은 옵션 선택
slice(2) %>%
ungroup() %>%
arrange(desc(BAS_DD)) %>%
left_join(idx_data, by="BAS_DD") %>%
left_join(mmf_raw, by="BAS_DD") %>%
# 옵션 보유기간 및 단기금리(MMF) 계산
mutate(DIFF_DAY=as.integer(ymd(lag(BAS_DD))-ymd(BAS_DD)),
MMF=MMF*0.01) %>%

```

```

# 옵션 보유기간에 해당하는 시장기대변동성 계산(V-K200활용)
mutate(VOL=LAG_VK200*sqrt(DIFF_DAY)/sqrt(365)/100) %>%
# 변동성에 따른 옵션 행사가격 상단(2.5단위 올림) 및 하단(2.5단위 내림) 계산
mutate(TARGET_C_K=ceiling(KOSPI200*(1+VOL)*0.4)/0.4,
       TARGET_P_K=floor(KOSPI200*(1-VOL)*0.4)/0.4) %>%
# drop_na(DIFF_DAY) %>%
select(BAS_DD,VOL,DIFF_DAY,MMF,KOSPI200,LAG_VK200,PRIOR,TARGET_C_K,TARGET_P_K)

# 3-2. 포트폴리오 구성 : VW양매도 포트폴리오 거래대상 옵션 정보 생성
target_options <- target_info %>%
  select(BAS_DD, PRIOR, TARGET_C_K, TARGET_P_K) %>%
  pivot_longer(cols=c("TARGET_C_K","TARGET_P_K"),names_to = "GRP", values_to = "EXER_PRC") %>%
  mutate(RGHT=substr(GRP,8,8)) %>%
  left_join(options,by=c("BAS_DD","PRIOR","RGHT","EXER_PRC")) %>%
  select(BAS_DD,GRP,PRICE,ACC_TRDVOL) %>%
  pivot_wider(names_from = "GRP", values_from = c("PRICE","ACC_TRDVOL"))

# 원금 100억원 가정
cash = 10000*10000*100

# 4. 포트폴리오 구현 : VW양매도 전략을 구현하여 일자별 손익 계산
portfolio <- target_info %>%
  left_join(target_options,by="BAS_DD") %>%
  arrange(BAS_DD) %>%
# 원금에 따른 옵션 매도수량 계산
  mutate(SELL_AMT=cash/250000/KOSPI200) %>%
# 옵션 프리미엄(매도수익) 계산
  mutate(PREMIUM=SELL_AMT*(PRICE_TARGET_C_K+PRICE_TARGET_P_K)*250000) %>%
# 원금 및 프리미엄의 MMF 이자수익 및 권리행사손실 계산
  mutate(INTEREST=(cash+replace_na(PREMIUM,0))*MMF*DIFF_DAY/365,
        EXERCISE=(if_else(lead(KOSPI200)-TARGET_C_K>0,lead(KOSPI200)-TARGET_C_K,0) +
                  if_else(TARGET_P_K-lead(KOSPI200)>0,TARGET_P_K-lead(KOSPI200),0))*250000*0.01),
# 주단위 포트폴리오 손익 계산(원금 100억 가정, 당일 투자시 다음주 실현손익을 의미)
  mutate(REVENUE=PREMIUM+INTEREST-EXERCISE) %>%
# 거래대상 옵션이 상장되어있지 않은 경우, MMF수익만 고려
  mutate(REVENUE;if_else(is.na(REVENUE),INTEREST,REVENUE)) %>%
  mutate(RATE=REVENUE/cash,
        Group="Vol-adj. Weekly") # 주단위 포트폴리오 수익률

```

```

# 5. 비교군 포트폴리오 생성 : 월별 5% OTM 양매도 전략
options2 <- options %>%
  filter(PROD=="코스피200")

# 옵션 만기일(매달) 생성
target_date2 <- options2 %>%
  distinct(.,BAS_DD, PRIOR) %>%
  arrange(PRIOR, desc(BAS_DD)) %>%
  group_by(PRIOR) %>%
  slice(1) %>%
  ungroup() %>%
  distinct(BAS_DD) %>%
  arrange(desc(BAS_DD)) %>%
  filter(BAS_DD<20250411)

# 일반 양매도 포트폴리오 기본 정보 생성
target_info2 <- target_date2 %>%
  left_join(options2, by="BAS_DD") %>%
  distinct(BAS_DD,PRIOR) %>%
  arrange(desc(BAS_DD),PRIOR) %>%
  group_by(BAS_DD) %>%
  # 만기가 도래하는 옵션을 제외하고 우선순위가 높은 옵션 선택
  slice(2) %>%
  ungroup() %>%
  arrange(desc(BAS_DD)) %>%
  left_join(idx_data, by="BAS_DD") %>%
  left_join(mmf_raw, by="BAS_DD") %>%
  # 옵션 보유기간 및 단기금리(MMF) 계산
  mutate(DIFF_DAY=as.integer(ymd(lag(BAS_DD))-ymd(BAS_DD)),
         MMF=MMF*0.01,
         VOL=0.05) %>%
  # 변동성에 따른 옵션 행사가격 상단(2.5단위 올림) 및 하단(2.5단위 내림) 계산
  mutate(TARGET_C_K=ceiling(KOSPI200*(1+VOL)*0.4)/0.4,
         TARGET_P_K=floor(KOSPI200*(1-VOL)*0.4)/0.4) %>%
  mutate(DIFF_DAY=if_else(is.na(DIFF_DAY),28,DIFF_DAY)) %>%
  select(BAS_DD,VOL,DIFF_DAY,MMF,KOSPI200,LAG_VK200,PRIOR,TARGET_C_K,TARGET_P_K)

# 일반 양매도 포트폴리오 거래대상 옵션
target_options2 <- target_info2 %>%

```

```

select(BAS_DD, PRIOR, TARGET_C_K, TARGET_P_K) %>%
pivot_longer(cols=c("TARGET_C_K","TARGET_P_K"),names_to = "GRP", values_to = "EXER_PRC") %>%
mutate(RGHT=substr(GRP,8,8)) %>%
left_join(options2,by=c("BAS_DD","PRIOR","RGHT","EXER_PRC")) %>%
select(BAS_DD,GRP,PRICE,ACC_TRDVOL) %>%
pivot_wider(names_from = "GRP", values_from = c("PRICE","ACC_TRDVOL"))

# 일반 양매도 포트폴리오 구현
portfolio2 <- target_info2 %>%
left_join(target_options2,by="BAS_DD") %>%
arrange(BAS_DD) %>%
# 원금에 따른 옵션 매도수량 계산
mutate(SELL_AMT=cash/250000/KOSPI200) %>%
# 옵션 프리미엄(매도수익) 계산
mutate(PREMIUM=SELL_AMT*(PRICE_TARGET_C_K+PRICE_TARGET_P_K)*250000) %>%
# 원금 및 프리미엄의 MMF 이자수익 및 권리행사손실 계산
mutate(INTEREST=(cash+replace_na(PREMIUM,0))*MMF*DIFF_DAY/365,
EXERCISE=(if_else(lead(KOSPI200)-TARGET_C_K>0,lead(KOSPI200)-TARGET_C_K,0)-
if_else(TARGET_P_K-lead(KOSPI200)>0,TARGET_P_K-lead(KOSPI200),0))*250000*
mutate(EXERCISE;if_else(is.na(EXERCISE),0,EXERCISE)) %>%
# 주단위 포트폴리오 손익 계산(원금 100억 가정, 당일 투자시 다음주 실현손익을 의미)
mutate(REVENUE=PREMIUM+INTEREST-EXERCISE) %>%
# 거래대상 옵션이 상장되어있지 않은 경우, MMF수익만 고려
mutate(REVENUE;if_else(is.na(REVENUE),INTEREST,REVENUE)) %>%
mutate(RATE=REVENUE/cash,
Group="Monthly") # 주단위 포트폴리오 수익률

portfolio <- portfolio %>% filter(BAS_DD!=20250410)
portfolio2 <- portfolio2 %>% filter(BAS_DD!=20250410)
portfolio3 <- portfolio %>%
summarise(across(where(is.numeric), sum)) %>%
mutate(Group="Vol-adj. Weekly Sum")

# 요약표 4 : 검증 자료
table4 <- portfolio %>%
union_all(portfolio2) %>%
union_all(portfolio3) %>%
group_by(BAS_DD, Group) %>%

```

```

summarise(Premium = round(mean(PREMIUM)/10000,0),
           Interest = round(mean(INTEREST)/10000,0),
           Loss = round(mean(EXERCISE)/10000,0),
           Revenue = round(mean(REVENUE)/10000,0),
           Return = round(prod(1 + RATE)-1,2),
           .groups = "drop")

# 그래프 4 : 검증기간중 코스피200지수 추이

graph4 <- ggplot(idx_raw, aes(x = ymd(BAS_DD))) +
  geom_line(aes(y = KOSPI200, color = "KOSPI200"), size = 1.2) +
  geom_line(aes(y = (VKOSPI200 - 19) / 26 * 60 + 300, color = "VKOSPI200"), size = 1.2) +
  geom_vline(xintercept = ymd(c("20250313", "20250320", "20250327", "20250403","20250410")), linetype = "dotted", color = "grey40") +
  scale_y_continuous(name = "KOSPI200",limits = c(300, 360),
                     sec.axis = sec_axis(
                       ~ (. - 300) / 60 * 26 + 19, name = "VKOSPI200")) +
  scale_color_manual(values = c("KOSPI200" = "blue", "VKOSPI200" = "red"),
                     guide = guide_legend(direction = "horizontal")) +
  labs(title = "Kospi200 & V-Kospi200 indices",x = NULL,color = NULL) +
  theme_minimal() +
  scale_x_date(date_breaks = "3 days", date_labels = "%m-%d") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = c(0.5, 0.95))

```

Part IV

장외파생상품 기초 실무('25 봄)

장외파생상품 기초 실무

Part V

금융윤리와 사회책임('25 봄)

금융윤리와 사회책임