28회 ADP 복원

packages

In [1]:
```
import pandas as pd
import math
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter

from IPython.display import display , HTML
display(HTML("<style>.container {width :95% !important;}</style>"))

from scipy.stats import chi2_contingency
import numpy as np

from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler

from imblearn.over_sampling import SMOTE


from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from lightgbm import LGBMClassifier

from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score , classification_report

import warnings
warnings.filterwarnings('ignore')

from lightgbm import LGBMClassifier
from sklearn.ensemble import VotingClassifier

from scipy import stats
import time
# from lifelines import KaplanMeierFitter
# from lifelines.statistics import logrank_test
from scipy.stats import mannwhitneyu
# import pingouin as pg
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-55d020f379fc> in <module>
     19 from sklearn.ensemble import RandomForestClassifier
     20 from sklearn.neural_network import MLPClassifier
---> 21 from lightgbm import LGBMClassifier
     22
     23 from sklearn.model_selection import train_test_split

ModuleNotFoundError: No module named 'lightgbm'
```

기계학습 (50점)

데이터 설명

- 데이터 출처 : https://www.kaggle.com/datasets/dipam7/student-grade-prediction?resource=download 후처리
- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p1.csv
- 데이터 설명 : 학생의 다양한 주변 환경에 따른 결석 등급 (absences)
  - sex : 성별 (F : 여성 / M :남성)
  - age : 나이
  - pstatus : 부모와 동거 유무 (T : 동거중 , A : 별거)
  - medu : 어머니 교육(0 - 없음, 1 - 초등 교육(4학년), 2 - 5~9학년, 3 - 중등 교육 또는 4 - 고등 교육)
  - fedu : 아버지 교육(0 - 없음, 1 - 초등 교육(4학년), 2 - 5 - 9학년, 3 - 중등 교육 또는 4 - 고등 교육)
  - guardian : 학생의 보호자
  - traveltime : 집에서 학교까지 이동 시간(1 - <15분, 2 - 15 - 30분, 3 - 30분 - 1시간, 또는 4 - >1시간)
  - studytime : 주간 학습 시간(1 - <2시간, 2 - 2 - 5시간, 3 - 5 - 10시간 또는 4 - >10시간)
  - failures : 과거 클래스 실패 수(n if 1<=n<3, 그렇지 않으면 4)
  - freetime : 방과 후 자유 시간(숫자: 1 - 매우 낮음에서 5 - 매우 높음)
  - famrel : 가족 관계의 질(숫자: 1 - 매우 나쁨에서 5 - 훌륭함)
  - absences : 학교 결석 횟수등급 (0~5, 높은 숫자일수록 많은 결석 횟수)

In [2]:
```
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p1.csv")
df.head()
```

Out[2]:

	sex	age	pstatus	medu	fedu	guardian	traveltime	studytime	failures	freetime	famrel	absences
0	F	18	A	4	4	mother	2	2	0	3	4	2
1	F	17	T	1	1	father	1	2	0	3	5	1
2	F	15	T	1	1	mother	1	2	3	3	4	3
3	F	15	T	4	2	mother	1	3	0	2	3	0
4	F	16	T	3	3	father	1	2	0	3	4	1

1-1. EDA를 진행하고 (+시각화) , 차원축소의 필요성이 있는지 확인 (5점)

In [3]:
```
print('- 데이터 샘플 확인')
display(df.head(3))
```

```
print('- 데이터 유형 확인')
display(df.info())
print('- 데이터 기초통계량 확인')
display(df.describe())
print('- 데이터 결측 확인')
display(df.isna().sum()[df.isna().sum()>0])
print('- 연속형 데이터 boxplot')

# 연속형 데이터만 추출
df_num = df.select_dtypes(exclude = 'object')

n_col = 5
n_row = math.ceil(len(df_num.columns)/n_col)

fig, axs = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*2))

for r in range(0, n_row):
    for c in range(0, n_col):
        i = r*n_col + c
        if i < len(df_num.columns):
            sns.boxplot(y = list(df_num.columns)[i], data = df, ax = axs[r][c])
plt.tight_layout()
plt.show()

print('- 데이터 분포 시각화')
# 데이터 분포 확인
def make_plot(df):
    plt.figure(figsize = (20, 10))
    for i, col in enumerate(df.select_dtypes(exclude=object).columns, start=1):
        plt.subplot(3, 4, i)
        plt.title(f'{col} Count')
        plt.hist(df[col])
        start_num = i+1

    for i, col in enumerate(df.select_dtypes(include=object).columns, start=start_num):
        plt.subplot(3, 4, i)
        plt.title(f'{col} Count')
        sns.countplot(data = df, x = col)
    plt.tight_layout()
    plt.show()

make_plot(df)


print('- 상관관계 확인')
df_cor = df_num.corr(method = 'pearson')
sns.heatmap(df_cor,
            xticklabels = df_cor.columns,
            yticklabels = df_cor.columns,
            cmap = 'RdBu_r',
            annot = True, linewidth = 3)
plt.show()

#print('- 이상치 처리')
#del_idx = []
#del_idx.extend(list(df[df.]))
```

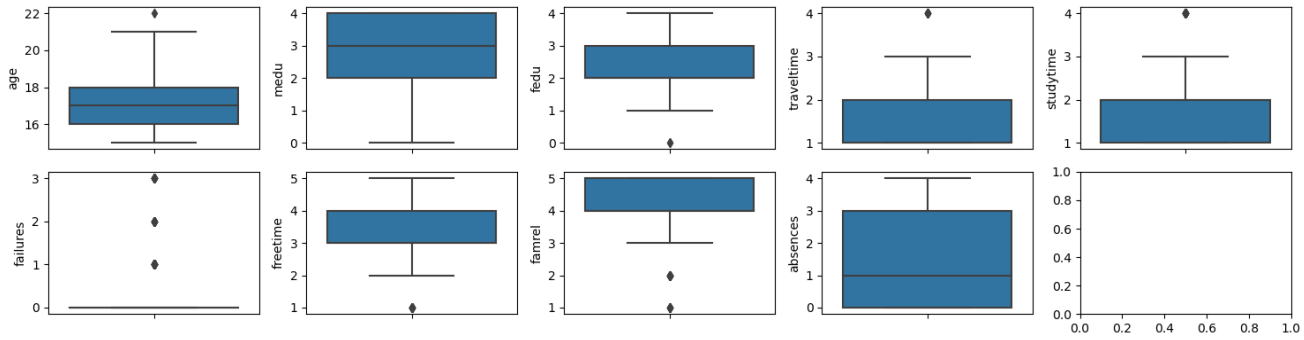
- 데이터 샘플 확인

	sex	age	pstatus	medu	fedu	guardian	traveltime	studytime	failures	freetime	famrel	absences
0	F	18	A	4	4	mother	2	2	0	3	4	2
1	F	17	T	1	1	father	1	2	0	3	5	1
2	F	15	T	1	1	mother	1	2	3	3	4	3

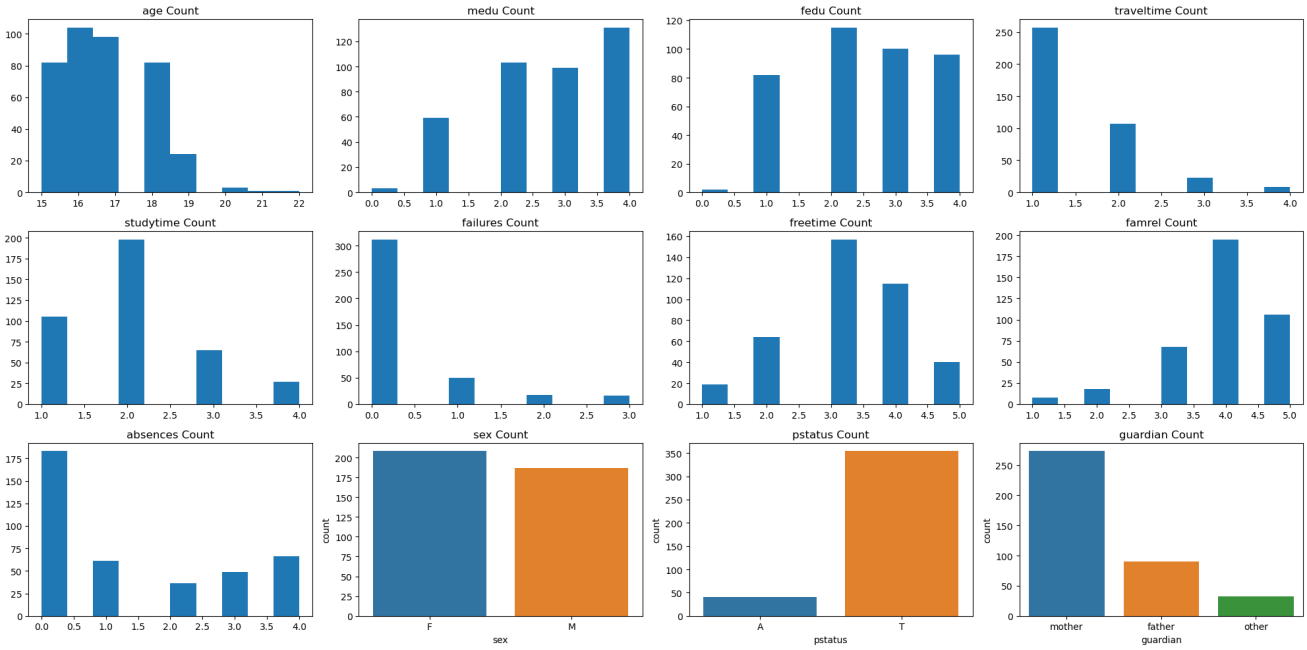
```
- 데이터 유형 확인
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   sex         395 non-null    object
 1   age         395 non-null    int64
 2   pstatus     395 non-null    object
 3   medu        395 non-null    int64
 4   fedu        395 non-null    int64
 5   guardian    395 non-null    object
 6   traveltime  395 non-null    int64
 7   studytime   395 non-null    int64
 8   failures    395 non-null    int64
 9   freetime    395 non-null    int64
 10  famrel      395 non-null    int64
 11  absences    395 non-null    int64
dtypes: int64(9), object(3)
memory usage: 37.2+ KB
None
- 데이터 기초통계량 확인
```

	age	medu	fedu	traveltime	studytime	failures	freetime	famrel	absences
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.235443	3.944304	1.377215
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.998862	0.896659	1.555076
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	3.000000	4.000000	0.000000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	3.000000	4.000000	1.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	4.000000	5.000000	3.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	4.000000

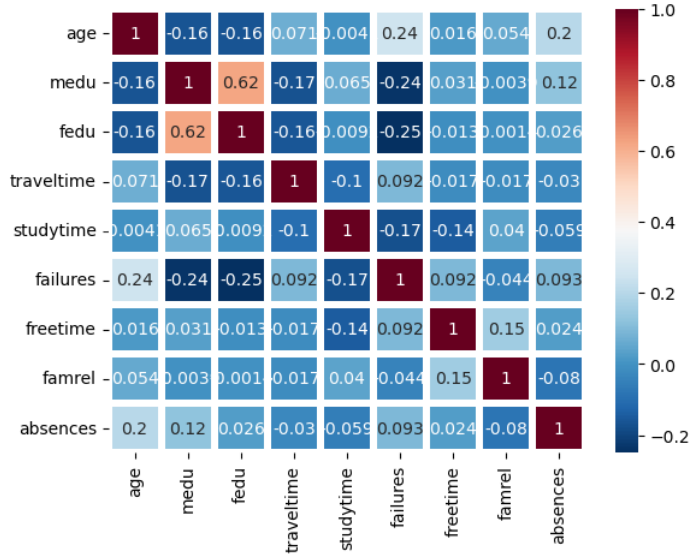
```
- 데이터 결측 확인
Series([], dtype: int64)
- 연속형 데이터 boxplot
```



- 데이터 분포 시각화



- 상관관계 확인



[답안] (34분)

- 데이터는 3개의 범주형 변수, 9개의 연속형변수로 이루어져 있으며 이상치나 결측치는 없는 것으로 보인다.
- 상관관계는 medu와 fedu가 양의 상관성이 크게 보이고 그 외의 변수는 상관성이 보이지 않는다. 그래도 차원축소를 해볼만은 할 것 같다.
- 타겟변수인 결석횟수가 0으로 치우쳐져있어서 모델을 만들기 전 데이터 불균형 문제를 해결하는 것이 좋다.

1-2. 데이터 품질 개선을 위한 방법이 있는지 찾고 데이터셋을 재생성하라 (5점)

In [70]:

```
# 데이터 품질을 개선시키기 위한 방법으로는 이상치나 결측치를 보완하는 방법, 차원을 축소를 통해 다중공선성 문제를 피하는 방법 등이 있다. 이 데이터는 이상치

print('- 차원축소')
x_col = list(df.columns)
x_col.remove('absences')
df_x = df[x_col]

# 인코딩
from sklearn.preprocessing import OneHotEncoder
df_x_enc = df_x.select_dtypes('object').copy()
enc = OneHotEncoder(sparse = False).fit(df_x_enc)
df_x_enc = pd.DataFrame(enc.transform(df_x_enc), columns = enc.get_feature_names_out())
```

```
# 스케일링
from sklearn.preprocessing import StandardScaler
df_x_num = df_x.select_dtypes(exclude='object').copy()
ss = StandardScaler().fit(df_x_num)
df_x_num = pd.DataFrame(ss.transform(df_x_num), columns = df_x_num.columns)

df_x2 = pd.concat([df_x_num, df_x_enc], axis = 1)

# PCA적합
model = PCA(n_components=df_x2.shape[1], svd_solver='auto')
model.fit(df_x2)

# 고유값 요약
e_value = pd.DataFrame({'고유값':model.explained_variance_, '기여율':model.explained_variance_ratio_},
                       index = ['comp%s'%i for i in range(1, df_x2.shape[1]+1)])
e_value['누적기여율'] = e_value['기여율'].cumsum()
print('고유값 요약:Wn', e_value, 'Wn')

print('comp1~8가 전체 분산의 90%를 설명하고 있으므로 n_components를 8로 하여 다시 만든다.')
# PCA 재적합
best_dim = 8
model = PCA(n_components=best_dim, svd_solver = 'auto')
model.fit(df_x2)
df_pca_x = pd.DataFrame(model.fit_transform(df_x2), columns = ['comp%s'%i for i in range(1, best_dim+1)])
df_pca = pd.concat([df_pca_x, df['absences']], axis = 1)
print('Wnpca 결과: ')
display(df_pca.head(3))

# 고유벡터 요약
# e_vector = pd.DataFrame(model.components_, index = ['comp%s'%i for i in range(1, best_dim+1)], columns = df_x2.columns)
# print('고유벡터 요약:Wn', e_vector.iloc[:, :5], 'Wn')

print('- 오버샘플링을 통해 데이터 불균형 문제 해결')
print('오버샘플링 전 건수 ', Counter(df['absences']))
smote = SMOTE(sampling_strategy = 'auto')
df_smote_x, df_smote_y = smote.fit_resample(df_pca_x, df['absences'])
print('오버샘플링 후 건수 ', Counter(df_smote_y))
```

```
- 차원축소
고유값 요약:
                    고유값            기여율      누적기여율
comp1    1.992887e+00  2.172865e-01  0.217286
comp2    1.318117e+00  1.437156e-01  0.361002
comp3    1.115034e+00  1.215733e-01  0.482575
comp4    1.000125e+00  1.090447e-01  0.591620
comp5    9.187687e-01  1.001743e-01  0.691794
comp6    7.655102e-01  8.346435e-02  0.775259
comp7    6.803603e-01  7.418038e-02  0.849439
comp8    4.401223e-01  4.798698e-02  0.897426
comp9    3.977288e-01  4.336477e-02  0.940791
comp10   2.802794e-01  3.055915e-02  0.971350
comp11   1.806252e-01  1.969375e-02  0.991044
comp12   8.214411e-02  8.956255e-03  1.000000
comp13   2.297283e-32  2.504751e-33  1.000000
comp14   9.613731e-33  1.048195e-33  1.000000
comp15   2.743369e-33  2.991123e-34  1.000000

comp1~8가 전체 분산의 90%를 설명하고 있으므로 n_components를 8로 하여 다시 만든다.

pca 결과:
```

	comp1	comp2	comp3	comp4	comp5	comp6	comp7	comp8	absences
0	-1.119243	-0.159530	0.135057	0.922367	1.688906	0.084629	-0.339341	0.073825	2
1	1.403234	0.776656	-1.330906	-0.806900	-0.649846	-1.001180	-0.458269	-0.873098	1
2	2.560491	0.223636	0.290734	0.220254	-2.017483	-0.398777	2.967237	0.024972	3

```
- 오버샘플링을 통해 데이터 불균형 문제 해결
오버샘플링 전 건수  Counter({0: 183, 4: 66, 1: 61, 3: 49, 2: 36})
오버샘플링 후 건수  Counter({2: 183, 1: 183, 3: 183, 0: 183, 4: 183})
```

[답안] (16분)

- 데이터 품질을 개선시키기 위한 방법으로는 이상치나 결측치를 보완하는 방법, 차원을 축소를 통해 다중공선성 문제를 피하는 방법, 데이터샘플링을 통해 불균형문제를 피하는 법 등이 있다. 이 데이터는 이상치나 결측치가 없는 것으로 판단되어 차원축소와 데이터 오버샘플링만 시행한다.
- 차원축소는 pca를 적용했다. 전체 변수의 수만큼 pca를 수행했을때 comp1~8이 전체 분산의 90%를 설명하고 있으므로 n_components를 8로 하여 만들었다.
- 오버샘플링은 SMOTE를 적용했으며 다수 클래스의 수에 맞게 데이터의 건수가 증가했다.

1-3. 1.2에서 제시한 방법이 데이터 과적합이 된다는 가정하에 어떻게 해결할 수 있을지 2가지 개선안 제시, 각방법들의 장단점 기술 (10점)

[답안]

- 과적합을 방지하기 위해서는 교차검증, 규제, 드롭아웃 등의 방법이 있다.
- 교차검증은 학습 데이터를 여러 부분으로 나누어 여러 파라미터 조건 하에 학습하여 교차 검증을 함으로써 최적화된 일반화 모델을 만들 수 있다는 장점이 있는 반면 계산 비용이 높고 큰 데이터셋에 대해 시간 소모가 클 수 있다는 단점이 있다.
- 규제는 l1, l2등의 규제를 통해 모델의 가중치를 제한함으로써 과적합을 방지할 수 있다. 예측의 안정성과 l1의 경우 특성을 선택하여 모델의 간결성을 높일 수 있는 장점이 있는 반면 과도한 규제는 모델을 단순하게 만들어 정확도가 낮을 수 있다는 단점이 있다.

2-1. 1-2 데이터셋을 기준으로 random forest, neural network , lightgbm 3가지 방식으로 학교 결석 횟수등급을 예측하는 모델을 만들어라, f1 score로 모델을 평가하라 (5점)

In [72]:

```
print('데이터 분할')
x_train, x_test, y_train, y_test = train_test_split(df_smote_x, df_smote_y, random_state=123, test_size = 0.3)

rf = RandomForestClassifier(n_estimators = 500)
model_rf = rf.fit(np.array(x_train), y_train)
pred_rf = model_rf.predict(np.array(x_test))

nn = MLPClassifier()
model_nn = nn.fit(np.array(x_train), y_train)
```

```
pred_nn = model_nn.predict(np.array(x_test))

lgbm = LGBMClassifier(n_estimators = 5, verbose=0, max_depth = 2)
model_lgbm = lgbm.fit(np.array(x_train), y_train)
pred_lgbm = model_lgbm.predict(np.array(x_test))

f1_rf = f1_score(y_test, pred_rf, average = 'micro')
f1_nn = f1_score(y_test, pred_nn, average = 'micro')
f1_lgbm = f1_score(y_test, pred_lgbm, average = 'micro')

print('f1_score 평가 결과 \n랜덤포레스트 : {:.3f}\n뉴럴네트워트: {:.3f}\nlightgbm : {:.3f}'.format(f1_rf, f1_nn, f1_lgbm))
```

```
데이터 분할
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
f1_score 평가 결과
랜덤포레스트 : 0.702
뉴럴네트워트: 0.487
lightgbm : 0.320
```

[답안]

- 1-2에서 pca를 적용한 데이터로 랜덤포레스트와 뉴럴네트워크, lightgbm 세 모델을 적용하고 f1 score로 비교하였다. 그 결과 랜덤포레스트 0.702 > 뉴럴네트워크 0.487 > lightgbm 0.32의 결과가 나왔다.

2-2. hard voting, soft voting에 대한 장단점을 설명하고 2-1의 3가지 모델로 구현하라. 두 방식의 f1-score를 비교하라 (10점)

In [73]:

```
# VotingClassifier를 사용하여 하드 보팅 구현
hard_voting = VotingClassifier(estimators=[('rf', model_rf), ('nn', model_nn), ('lgbm', model_lgbm)], voting='hard')
hard_voting.fit(x_train, y_train)
pred_hard = hard_voting.predict(x_test)

# VotingClassifier를 사용하여 소프트 보팅 구현
soft_voting = VotingClassifier(estimators=[('rf', model_rf), ('nn', model_nn), ('lgbm', model_lgbm)], voting='soft')
soft_voting.fit(x_train, y_train)
pred_soft = soft_voting.predict(x_test)

# 평가
f1_hard = f1_score(y_test, pred_hard, average = 'micro')
f1_soft = f1_score(y_test, pred_soft, average = 'micro')

print('f1_score 평가 결과 \nhard voting : {:.3f}\nsoft voting: {:.3f}'.format(f1_hard, f1_soft))
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
f1_score 평가 결과
hard voting : 0.549
soft voting: 0.604
```

[답안]

- hard voting은 각 모델의 예측값을 바탕으로 다수결 투표하는 방식으로 모든 개별 모델이 동일한 중요도를 갖고 있을때 효과적인 장점이 있으나 성능이 뛰어난 모델과 약한 모델의 가중치를 구분하지 못하다는 단점이 있다.
- soft voting은 예측값들의 평균 및 가중치 합을 사용하는 방식으로 불확실성을 더 잘 다룰 수 있지만 모든 모델의 예측 확률을 독립적으로 계산할 수 없는 경우 사용하기 어렵다는 단점이 있다.
- 두 voting을 적용한 결과 soft voting(0.604) > hard voting(0.549)로 soft_voting이 더 좋은 결과를 얻었다.

2-3. 총 5개 모델(RF, NN, LGBM, 하드보팅, 소프트보팅) 중 실시간 온라인 시스템에 가장 적합한 모델과 선정이유를 객관적으로 제시하라 (5점)

[답안]

- 5개의 모델에서 실시간 온라인 시스템에 가장 적합한 모델은 랜덤포레스트이다. 다른 모델과 성능차가 많이 나는데다가 성능이 가장 좋기 때문이다. nn, lgbm의 경우 랜덤포레스트에 비해 성능이 너무 떨어지기 때문에 하드보팅이나 소프트보팅으로 그 예측값을 넣는 것은 좋지 않다. 성능차가 크게 없었더라면 소프트 보팅을 선택했을 것이다.

3-1. 적정 모델과 선정 및 모델링 과정에서 추가적으로 고려해볼 만한 사항은? (5점)

[답안]

- grid search를 통해 하이퍼파라미터를 선택하는 과정을 추가하여 각 모델의 성능을 끌어올리면 좋을 것 같다.

3-2. 모델을 학교 시스템에 적용하여 활용하려한다. 모델 적용 및 운영과정에서 고려해볼 만한 사항? (5점)

[답안] (1시간 20분)

- 모델을 실제 업무에 적용할 때 성능도 중요하지만 데이터 수집과 비용, 시간이 중요하다. 데이터 수집단계에서 담당자의 업무로드가 클 수 있기 때문에 최대한 자동화하는 것이 좋고 또, 데이터수집부터 분류 결과까지의 시간이 어느정도 걸리는지 체크해서 효율적으로 계산비용이 적게 코드를 작성하는 것이 좋다.

통계 (50점)

4번 데이터

- 데이터 url : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p4_.csv

- status : 생존 여부 (death: 죽음 / event lost: 생존)
- company : 회사구분

In [286]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p4_.csv')
df.head()
```

Out[286]:

	time(month)	status	company
0	1	event lost	X
1	2	event lost	X
2	3	event lost	X
3	4	event lost	X
4	5	event lost	X

4-1 Kaplan Meier 방법 사용 생존분석 수행. 회사부품별 25, 35, 45 개월에서의 생존 확률 (소숫점 3자리 ,5점)

In [ ]:
```

```

4-2 두 회사간 생존시간 차이를 log-rank 방식으로 검정하시오. 가설설정, 통계량, 귀무가설 기각여부 판단(10점)

In [ ]:
```

```

5번 데이터

- 데이터 url: https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p5_.csv
- data형식
  - 한 유저가 시식 전 물건 구매의사 유,무와 시식 후 구매의사 유,무에 대한 응답을 나타낸 데이터
  - {시식전} *{구매의사 유 or 무}* _ *{시식후}* {구매의사 유 or 무}

In [2]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p5_.csv')
df.head()
```

Out[2]:

	data	userId
0	시식전_유__시식후_유	user_1
1	시식전_유__시식후_유	user_2
2	시식전_유__시식후_유	user_3
3	시식전_유__시식후_유	user_4
4	시식전_유__시식후_유	user_5

5-1 시식여부가 구매의사에 영향을 주는지 가설을 설정하시오(5점)

[답안]

- 귀무가설(H0): 시식여부가 구매의사에 영향을 주지 않는다.
- 연구가설(H1): 시식여부가 구매의사에 영향을 준다.

5-2 검정하고 결과를 분석하시오(5점)

In [10]:
```
from scipy.stats import chi2

# Off-diagonal elements
b = df[df.data == '시식전_유__시식후_무'].shape[0]  # 시식 전 '유' & 시식 후 '무'
c = df[df.data == '시식전_무__시식후_유'].shape[0] # 시식 전 '무' & 시식 후 '유'

# McNemar test statistic
chi_squared = ((b - c) ** 2) / (b + c)

# P-value from chi-squared distribution
p_value = chi2.sf(chi_squared, 1)  # 1 degree of freedom

# 결과 출력
print("Chi-squared:", chi_squared)
print("P-value:", p_value)
```

```
Chi-squared: 3.24
P-value: 0.07186063822585143
```

[답안]

- 유의확률이 0.07로 유의수준 0.05보다 크므로 귀무가설을 채택한다.

6번 데이터

- 데이터 url: https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p6.csv

6-1 A,B 지역 학생의 점수에 차이가 있는지 가설을 설정하고 정하시오 (10점)

In [12]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p6.csv')
display(df.head(3))
# 결측 제거
df2 = df.dropna()
a = df2.loc[df2.school_name == 'A', 'score']
b = df2.loc[df2.school_name == 'B', 'score']


print(stats.bartlett(a, b))
print(stats.ttest_ind(a, b, equal_var=False))
```

	score	school_name	ID
0	91.0	A	1
1	NaN	A	2
2	NaN	A	3

```
BartlettResult(statistic=5.632798481244901, pvalue=0.01762746829155786)
TtestResult(statistic=-2.0511995199124358, pvalue=0.0635266327411075, df=11.604337033322173)
```

[답안]

- 귀무가설(H0): A, B지역 학생의 점수에 차이가 없다.
- 연구가설(H1): A, B지역 학생의 점수에 차이가 있다.
- 귀무가설(H0): A, B지역 학생의 점수가 등분산이다.
- 연구가설(H1): A, B지역 학생의 점수가 등분산이 아니다.
- 먼저 가설검정을 하기 전 데이터의 결측을 제거한 후 분산 검정을 수행했다. 그 결과 유의확률이 0.017로 유의수준이 0.05보다 작으므로 귀무가설을 기각한다. 즉 등분산이 아니다.
- 그런 후 독립표본 t검정을 실시한 결과 유의확률이 0.063으로 유의수준이 0.05보다 크므로 귀무가설을 기각하지 못한다. 즉 두 지역 학생의 점수차는 없다.

7번 데이터

- 데이터 출처 : https://www.kaggle.com/datasets/hangawqadir/erbil-heart-disease-dataset
- 데이터 url : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p7.csv
- 연령, 몸무게,콜레스테롤 수치 데이터

In [13]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/28/p7.csv')
df.head()
```

Out[13]:

	age	Cholesterol	weight
0	65	69.0	111.0
1	54	117.0	81.0
2	61	86.2	72.0
3	57	76.0	78.0
4	62	160.0	61.0

7-1 몸무게를 제어한다고 생각하고, 나이와 콜레스테롤 상관계수 및 유의확률 구하라(10점)

In [17]:
```
import pandas as pd
import statsmodels.api as sm

# 독립 변수 (나이, 콜레스테롤)에 상수항(intercept)을 추가함
X = sm.add_constant(df[['age', 'Cholesterol']])

# 다중 선형 회귀 모델
model = sm.OLS(df['weight'], X).fit()

print(model.summary())
print(np.sqrt(0.044))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 weight   R-squared:                       0.044
Model:                            OLS   Adj. R-squared:                  0.038
Method:                 Least Squares   F-statistic:                     7.574
Date:                Thu, 26 Oct 2023   Prob (F-statistic):           0.000608
Time:                        23:10:39   Log-Likelihood:                -1374.9
No. Observations:                 333   AIC:                             2756.
Df Residuals:                     330   BIC:                             2767.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          74.8953      4.455     16.813      0.000      66.132      83.658
age            -0.0361      0.059     -0.611      0.542      -0.152       0.080
Cholesterol     0.0819      0.022      3.716      0.000       0.039       0.125
==============================================================================
Omnibus:                       15.848   Durbin-Watson:                   2.033
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               17.569
Skew:                           0.471   Prob(JB):                     0.000153
Kurtosis:                       3.617   Cond. No.                         701.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
0.20976176963403032
```

- 나이와 콜레스테롤 상관계수는 결정계수의 제곱근인 0.209이고 유의확률은 0.000608이다.

7-2 상관계수를 유의수준 **0.05**하에서 검정하라 **(5점)**

In [ ]: