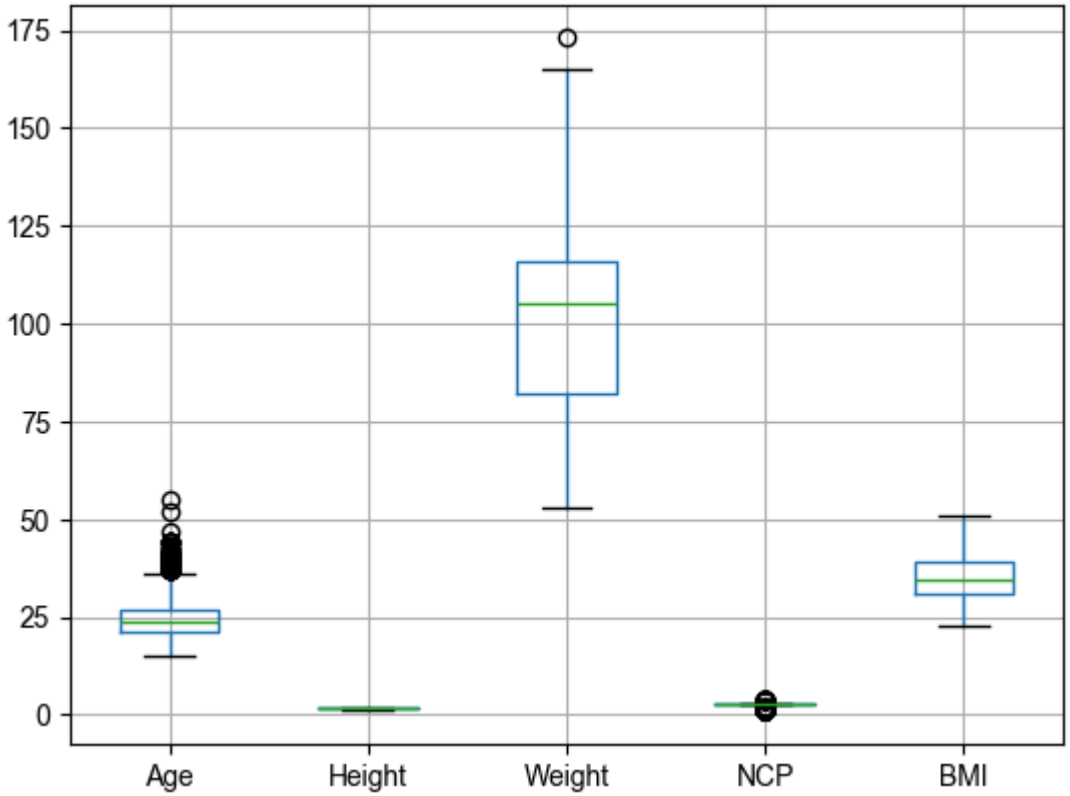
In [27]:
```
# 1. 데이터 설명 : 각 환자의 의료정보이다. NObeyesdad를 종속변수로 하는 분류모델을 만드려고 한다.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_1_obesity_v2.csv",index_col=0)
df.head(5)
```

Out[27]:

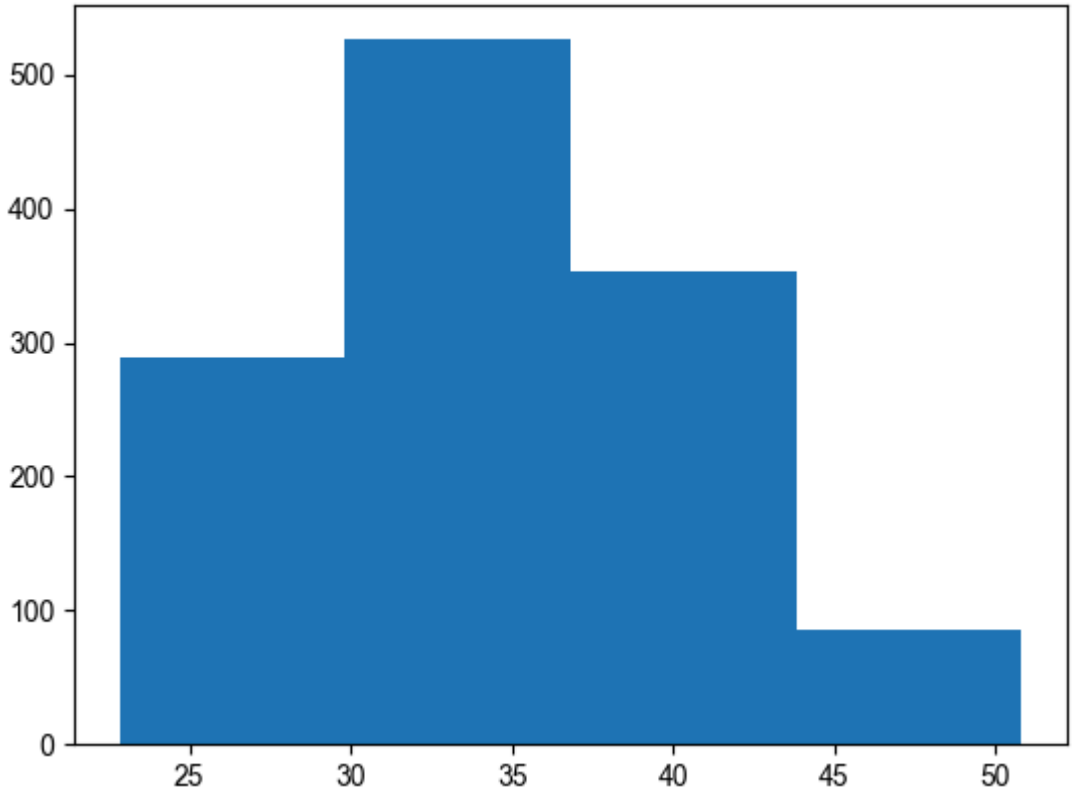
	id	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	
0	4	male	27	1.80	87	no	no	always	3	sometimes	no	between 1 and 2 l	no	2 to 4	0 to 2	fre
1	11	male	26	1.85	105	yes	yes	always	3	frequently	no	more than 2 l	no	2 to 4	>5	som
2	14	male	41	1.80	99	no	yes	sometimes	3	sometimes	no	between 1 and 2 l	no	2 to 4	3 to 5	fre
3	18	female	29	1.53	78	no	yes	sometimes	1	sometimes	no	between 1 and 2 l	no	0	0 to 2	
4	20	female	23	1.65	70	yes	no	sometimes	1	sometimes	no	between 1 and 2 l	no	0	0 to 2	som

In [28]:
```
# 1-1. EDA & 결측치 및 이상치를 판단하고 처리하라
#결측치 확인
df.isna().sum()
df[df["SCC"].isna()]
df.dropna(inplace=True)#SCC: Calories consumption monitoring - yes or no Null값 3개 존재해서, Null값 삭제진행
#이상치 확인
df_rev=df[df["Age"]<100]
df_rev.boxplot()
plt.show()
```



In [29]:
```
# 1-2. 데이터 전처리 기법 2가지를 설명하고 주어진 데이터에 적용시 어떤 효과가 있는지 설명하라
# [누락된 값 처리 (Handling Missing Values)]
# 장점:데이터의 유실을 방지하고 분석의 왜곡 감소(대체값을 사용하여 데이터의 손실을 최소화 가능)
# 단점: 잘못된 대체로 인해 모델의 정확성이 저하

# [이상치 탐지 및 처리 (Outlier Detection and Treatment)]
# 장점:이상치를 탐지하고 이를 처리하여 모델의 예측 개선
# 단점:이상치를 처리함에 따라 유용한 정보 손실 우려

# [정규화 (Normalization)]
# 장점:다양한 스케일의 데이터를 비교 가능
# 단점:데이터의 분포 왜곡

# [원핫 인코딩 (One-Hot Encoding)]
# 장점:범주형 데이터를 모델이 이해할 수 있는 형태로 변환
# 단점:고차원의 데이터로 인해 모델의 복잡성 증가
# [출처] [ADP 실기 기출 풀이] 제 31회 ADP 실기 기출(문제1,2,3)|작성자 mini_log-
```

In [30]:
```
plt.hist(df_rev["BMI"],bins=4)
plt.show()
```



In [31]:
```python
def make_newfeature(bmi):

    if bmi<30:
        return "저체중"
    elif bmi<37:
        return "슬림체형"
    elif bmi<44:
        return "근육돼지"
    else:
        return "과체중"

df_rev["new_feature"]=df_rev["BMI"].apply(make_newfeature)
```

```
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_22065/159885691.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_rev["new_feature"]=df_rev["BMI"].apply(make_newfeature)
```

In [32]:

```
# 2-1. 앙상블을 제외한 분류 모델 3가지 구축 및 결과 비교 및 설명하라
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

df_rev.drop(columns="id",inplace=True) #id Column 제거
df_rev["NCP"]=df_rev["NCP"].astype("object") #NCP object로 Type 변환

X=df_rev.drop(columns="NObeyesdad")
y=df_rev["NObeyesdad"]

#numeric값들 normalization 진행
scale_std=StandardScaler()
X["Age"]=scale_std.fit_transform(X[["Age"]])
X["Height"]=scale_std.fit_transform(X[["Height"]])
X["Weight"]=scale_std.fit_transform(X[["Weight"]])

X_rev=pd.get_dummies(X)
X_train, X_test, y_train, y_test=train_test_split(X_rev,y,test_size=0.2)

#LogisticRegression()
model_lr=LogisticRegression()#LogisticRegression(penalty='l2',)penalty='elasticnet'. Setting l1_ratio=0 is equivalent to using
model_lr.fit(X_train,y_train)
model_lr_pred=model_lr.predict(X_test)
print(classification_report(y_test,model_lr_pred))

#DecisionTreeClassifier()
model_DT=DecisionTreeClassifier()
model_DT.fit(X_train,y_train)
model_DT_pred=model_lr.predict(X_test)
print(classification_report(y_test,model_DT_pred))

#SVC()
model_SVC=SVC()
model_SVC.fit(X_train,y_train)
model_SVC_pred=model_lr.predict(X_test)
print(classification_report(y_test,model_SVC_pred))
```

```
                   precision    recall  f1-score   support

    obesity_type_i       0.99      0.99      0.99        75
   obesity_type_ii       0.95      0.97      0.96        62
  obesity_type_iii       0.98      0.98      0.98        57
overweight_level_i       1.00      0.98      0.99        57

          accuracy                           0.98       251
         macro avg       0.98      0.98      0.98       251
      weighted avg       0.98      0.98      0.98       251

                   precision    recall  f1-score   support

    obesity_type_i       0.99      0.99      0.99        75
   obesity_type_ii       0.95      0.97      0.96        62
  obesity_type_iii       0.98      0.98      0.98        57
overweight_level_i       1.00      0.98      0.99        57

          accuracy                           0.98       251
         macro avg       0.98      0.98      0.98       251
      weighted avg       0.98      0.98      0.98       251

                   precision    recall  f1-score   support

    obesity_type_i       0.99      0.99      0.99        75
   obesity_type_ii       0.95      0.97      0.96        62
  obesity_type_iii       0.98      0.98      0.98        57
overweight_level_i       1.00      0.98      0.99        57

          accuracy                           0.98       251
         macro avg       0.98      0.98      0.98       251
      weighted avg       0.98      0.98      0.98       251
```

```
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_22065/137481762.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_rev.drop(columns="id",inplace=True) #id Column 제거
/var/folders/hv/lqp1gn9n1ll0lbh2pfzn9pww0000gn/T/ipykernel_22065/137481762.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_rev["NCP"]=df_rev["NCP"].astype("object") #NCP object로 Type 변환
/opt/homebrew/Caskroom/miniforge/base/envs/general/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [33]:

```python
# 2-2 2-1에서 사용한 모델 중 하나를 골라 그리드 서치를 통해서 파라미터 튜닝 및 분류 모델 성능 평가 (precision ,recall)
from sklearn.model_selection import GridSearchCV

grid={"max_depth":[10,20,40],"criterion":["gini", "entropy"],'min_samples_split': [2, 5, 10]}

model_dt_grid=GridSearchCV(model_DT,param_grid=grid,cv=3)

model_dt_grid.fit(X_train,y_train)
model_dt_grid=model_dt_grid.best_estimator_
# model_DT_grid=DecisionTreeClassifier(criterion='entropy', max_depth=20)
model_dt_grid.fit(X_train,y_train)
model_dt_grid_pred=model_dt_grid.predict(X_test)
print(classification_report(y_test,model_dt_grid_pred))
```

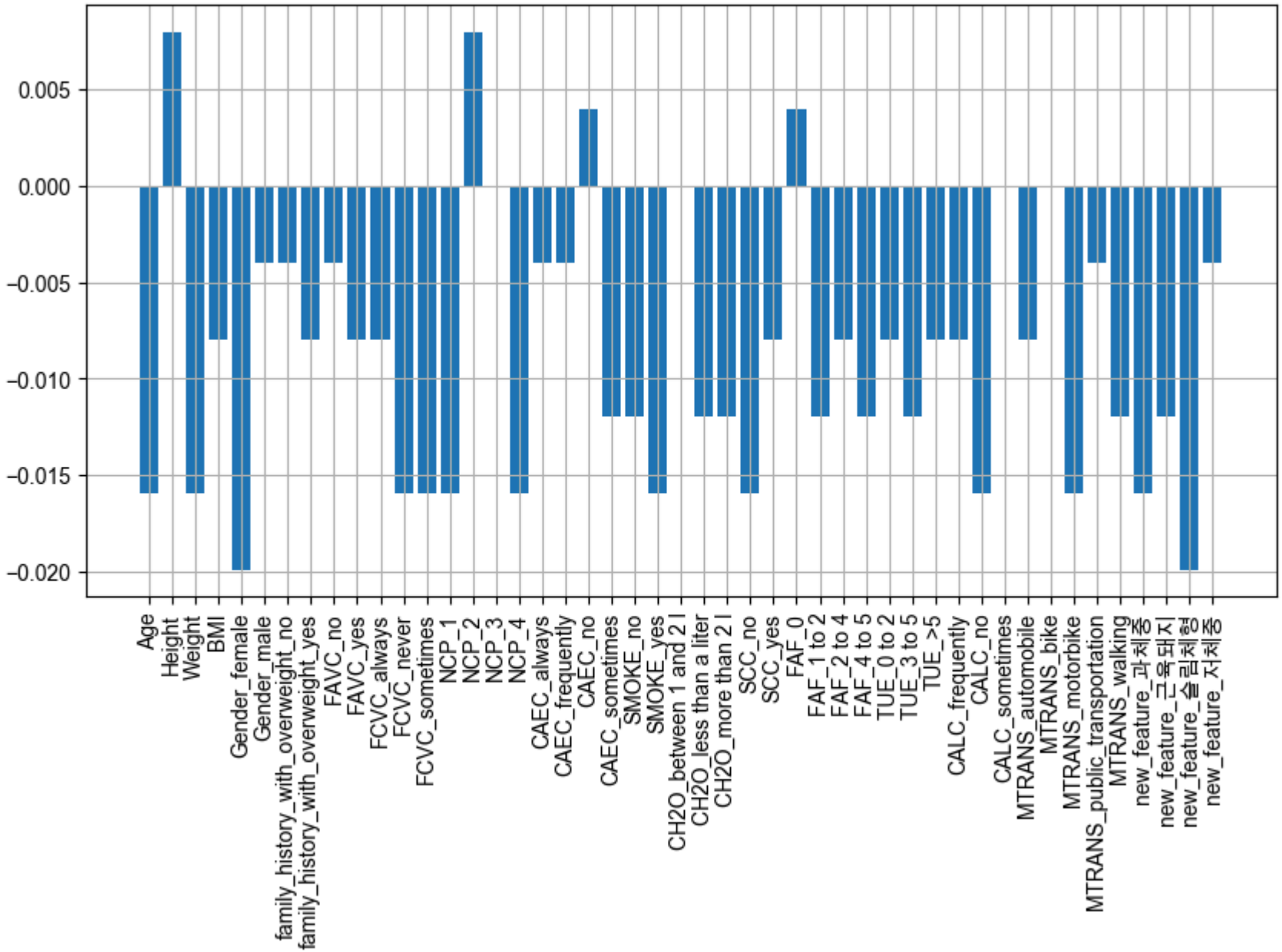
```
                    precision    recall  f1-score   support

      obesity_type_i       1.00      1.00      1.00        75
     obesity_type_ii       0.98      0.98      0.98        62
    obesity_type_iii       0.98      0.98      0.98        57
   overweight_level_i       1.00      1.00      1.00        57

            accuracy                           0.99       251
           macro avg       0.99      0.99      0.99       251
        weighted avg       0.99      0.99      0.99       251
```

In [34]:
```python
# 2-3 2-1의 3가지 모델을 soft voting을 이용하여 모델링 한 결과와 2-2과 비교하라
from sklearn.ensemble import VotingClassifier
model_DT.probability = True
model_lr.probability = True
model_SVC.probability = True

model_soft=VotingClassifier(estimators=[('lr', model_lr), ('DT', model_DT), ('SVC', model_SVC)],voting='soft')
model_soft.fit(X_train,y_train)

model_soft_pred=model_soft.predict(X_test)
print(classification_report(y_test,model_soft_pred))
```

```
                    precision    recall  f1-score   support

      obesity_type_i       1.00      1.00      1.00        75
     obesity_type_ii       0.98      0.98      0.98        62
    obesity_type_iii       0.98      0.98      0.98        57
   overweight_level_i       1.00      1.00      1.00        57

            accuracy                           0.99       251
           macro avg       0.99      0.99      0.99       251
        weighted avg       0.99      0.99      0.99       251
```

```
/opt/homebrew/Caskroom/miniforge/base/envs/general/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:465: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [35]:
```python
# 3-1 하나의 모델을 선정하여 Drop Column Importance 방식으로 각 변수 별로 전체 컬럼을 포함한 모델과의 recall 값의 차이를 계산하고 시각화 하라
from sklearn.metrics import recall_score
from matplotlib import font_manager, rc
font_path='/Library/Fonts/Arial Unicode.ttf'
font= font_manager.FontProperties(fname=font_path).get_name()
rc('font',family=font)
model_DT_recall=recall_score(y_test,model_DT_pred,average="micro")

recall_diff={}
for col in X_rev.columns:
    X_rev_drop=X_rev.drop(columns=col)
    X_train_drop, X_test_drop, y_train_drop, y_test_drop=train_test_split(X_rev_drop,y,test_size=0.2)

    model_DT.fit(X_train_drop,y_train_drop)
    pred_imsi=model_DT.predict(X_test_drop)
    recall_imsi=recall_score(y_test_drop,pred_imsi,average="micro")

    recall_diff[col]=model_DT_recall-recall_imsi

plt.figure(figsize=(10,5))
plt.bar(height=recall_diff.values(),x=recall_diff.keys())
plt.xticks(rotation=90)
plt.grid()
plt.show()
# [출처] [ADP 실기 기출 풀이] 제 31회 ADP 실기 기출(문제1,2,3)|작성자 mini_log-
```



In [36]:
```
# 4-1. 아래의 기준으로 전처리를 하여 적정 체중 여부 컬럼을 생성하고 BMI를 5단위로 구분하여 와 적정 체중여부에 대한 빈도 표를 만들어라

# (데이터 설명 : 중고등학생 건강검진 데이터)

df =pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_2_v2.csv')
df.head()
```

Out[36]:

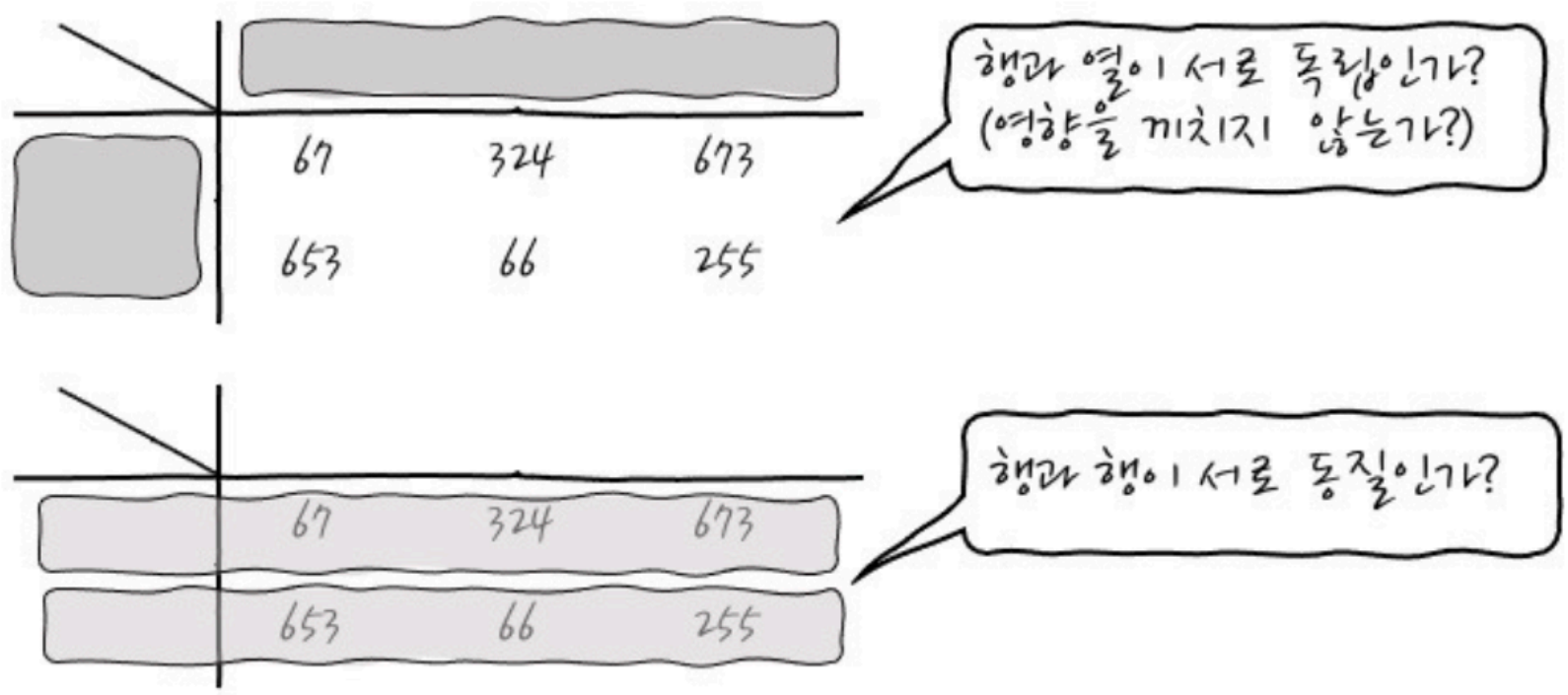
	ID	키	weight	생년월일	건강검진일	공학여부	채소섭취정도	아침식사여부	일주일운동시간	수면시간	성별
0	ID_4135	169.01	65.47	20041003	2020_11_15	1	2	1	4.4	8.3	남성
1	ID_3289	181.62	69.36	19970725	2014_11_20	0	3	0	4.4	6.9	남성
2	ID_1847	160.89	65.12	20020921	2020_01_28	1	1	1	1.7	9.6	여성
3	ID_4785	162.21	62.28	20020106	2018_09_27	1	4	0	5.1	6.8	남성
4	ID_5693	159.13	54.11	19980708	2015_03_03	0	4	1	0.3	8.5	여성

In [37]:

```
# Bmi = 몸무게(kg)/(키(m)**2)
df["BMI"]=df["weight"]/(df["키"]/100)**2

"""만나이 구하기 - 건강검진을 받았던 날을 기준으로 생년월일과 일수 차이가 16년 364일 이하인 경우 만 16세 그 이상의 경우 만 17로 분류하라 -
윤년 등은 고려하지 않는다. 햇수로 16년 + 일수로 364일이 기준이다"""

df["생년월일_rev"]=df["생년월일"].apply(lambda x: pd.to_datetime(x,format="%Y%m%d",errors="coerce"))
df["건강검진일_rev"]=df["건강검진일"].apply(lambda x: pd.to_datetime(x,format="%Y_%m_%d",errors="coerce"))
df["delta"]=df["건강검진일_rev"]-df["생년월일_rev"]
df["delta_rev"]=df["delta"].dt.days/365

def make_age(x):
    if x<=16.364:
        return "만16세"
    else:
        return "만17세"

df["만나이"]=df["delta_rev"].apply(make_age)

"""적정 체중 여부 (BMI가 아래 구간에 들어올 경우)
17세 남자 : 21.03이상 23.21 미만
17세 여자 : 20.03이상 22.39 미만
16세 남자 : 21.18이상 23.45 미만
16세 여자 : 19.61이상 21.74 미만"""

def make_bmi(data):

    if data["만나이"]=="만17세":
        if (data["성별"]=="남성")&(data["BMI"]>=21.03)&(data["BMI"]<23.21):
            return "적정"
        elif (data["성별"]=="여성")&(data["BMI"]>=20.03)&(data["BMI"]<22.39):
            return "적정"
        else:
            return "비적정"
```

```
    else:
        if (data["성별"]=="남성")&(data["BMI"]>=21.18)&(data["BMI"]<23.45):
            return "적정"
        elif (data["성별"]=="여성")&(data["BMI"]>=19.61)&(data["BMI"]<21.74):
            return "적정"
        else:
            return "비적정"

df["적정여부"]=df.apply(make_bmi,axis=1)
```

독립성과 동질성을 위해 교차표를 이용하는 손쉬운 방법을 정리하면 다음과 같습니다. 꽤나 유용하게 이용해 먹고 있습니다만.



In [38]:
```
# 4-2. 에서 구한 적정 체중 여부와 나머지 컬럼(공학여부, 아침식사여부, 일주일운동시간, 채소섭취정도, 수면시간, 성별) 이 독립적인지 통계적으로 확인하라
# 적정 체중 여부 vs 공학여부
# 귀무가설 (\(H_{0}\)): 두 변수는 서로 독립적이다 (관계가 없다).
# 대립가설 (\(H_{1}\)): 두 변수는 서로 독립적이지 않다 (관계가 있다).
```

```
import scipy.stats as stats
data=stats.chi2_contingency(pd.crosstab(df['적정여부'], [df['공학여부']]))
print(data.pvalue)
#pvalue : 0.03618078250755496

# 적정 체중 여부 vs 아침식사여부
data=stats.chi2_contingency(pd.crosstab(df['적정여부'], [df['아침식사여부']]))
print(data.pvalue)
#pvalue : 0.6419817098035392

# 적정 체중 여부 vs 성별
data=stats.chi2_contingency(pd.crosstab(df['적정여부'], [df['성별']]))
print(data.pvalue)
#pvalue : 0.37038572280812176
# [출처] [ADP 실기 기출 풀이] 제 31회 ADP 실기 기출(문제4,5,6,7)|작성자 mini_log-
```

```
0.03618078250755496
0.6419817098035392
0.37038572280812176
```

In [39]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df["적정여부_rev"]=le.fit_transform(df["적정여부"])

stats.pearsonr(df["적정여부_rev"],df["일주일운동시간"])
stats.pearsonr(df["적정여부_rev"],df["수면시간"])
```

Out[39]:
```
PearsonRResult(statistic=0.027188872624964916, pvalue=0.02494721828966266)
```

In [40]:
```
# 4-3. 4-2에서 유의한 변수들만 가지고 적정 체중 여부를 예측하는 모델을 구현하고 성능 평가 및 해석을 하라 (로지스틱회귀 ,xgb)
# 로지스틱회귀 모델은 오즈비를 구하라
# xgb의 경우 각 피쳐중요도를 확인하고 예측에 영향을 가장 미치는 변수를 확인하라
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from statsmodels.formula.api import ols

model_ols=ols(data=df,formula='적정여부_rev ~ 키+weight+공학여부+채소섭취정도+아침식사여부+일주일운동시간+수면시간+성별+만나이').fit()
model_ols.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	적정여부_rev	**R-squared:**	0.006
Model:	OLS	**Adj. R-squared:**	0.005
Method:	Least Squares	**F-statistic:**	4.748
Date:	Thu, 09 Oct 2025	**Prob (F-statistic):**	2.53e-06
Time:	11:43:47	**Log-Likelihood:**	-4848.5
No. Observations:	6801	**AIC:**	9717.
Df Residuals:	6791	**BIC:**	9785.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1583	0.168	0.941	0.347	-0.171	0.488
성별[T.여성]	0.0368	0.020	1.882	0.060	-0.002	0.075
만나이[T.만17세]	-0.0928	0.018	-5.135	0.000	-0.128	-0.057
키	0.0014	0.001	1.157	0.247	-0.001	0.004
weight	0.0026	0.002	1.701	0.089	-0.000	0.006
공학여부	0.0254	0.012	2.122	0.034	0.002	0.049
채소섭취정도	0.0023	0.004	0.530	0.596	-0.006	0.011
아침식사여부	-0.0038	0.012	-0.320	0.749	-0.027	0.020
일주일운동시간	0.0019	0.002	0.902	0.367	-0.002	0.006
수면시간	0.0096	0.004	2.320	0.020	0.001	0.018

Omnibus:	25124.007	**Durbin-Watson:**	1.969
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1107.814
Skew:	-0.277	**Prob(JB):**	2.76e-241
Kurtosis:	1.102	**Cond. No.**	4.92e+03

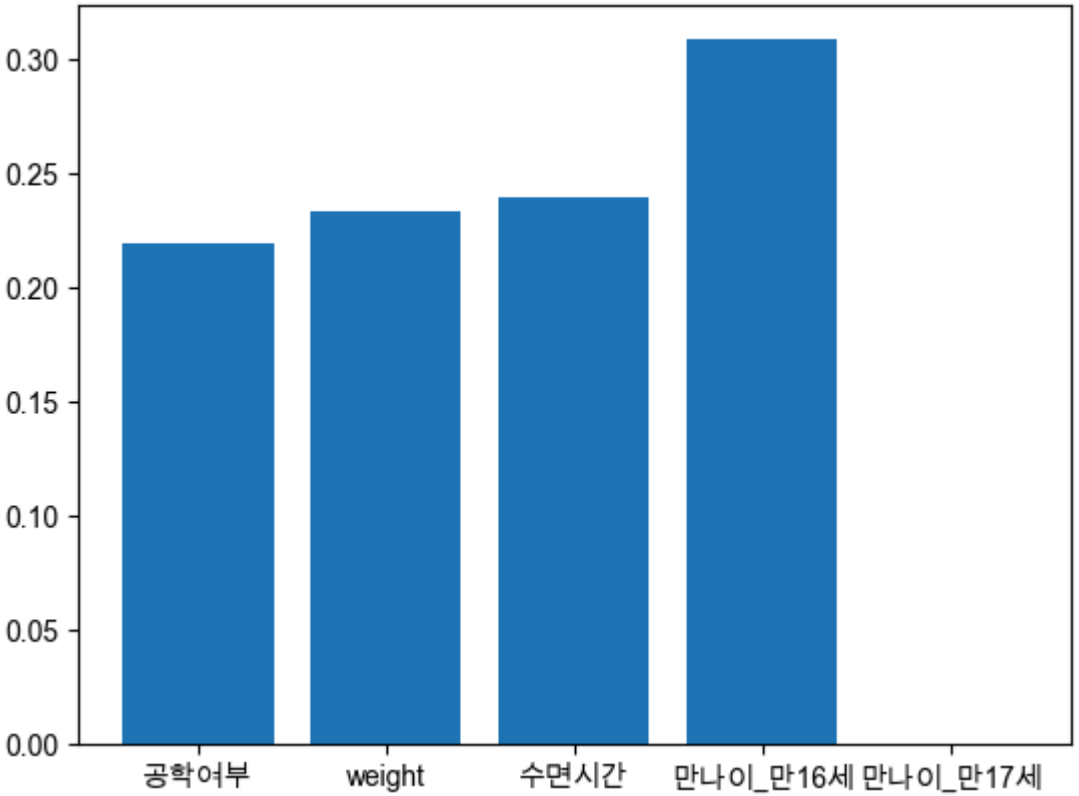
Notes:

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

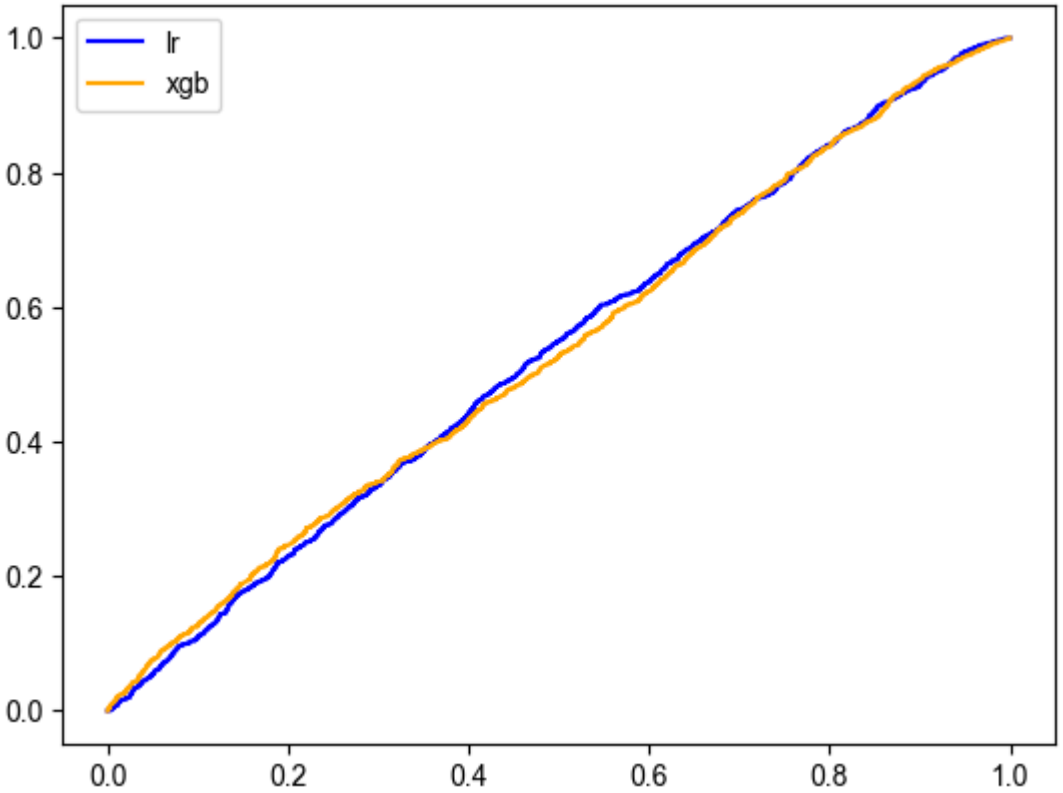
[2] The condition number is large, 4.92e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [41]:

```
from sklearn.model_selection import train_test_split
from matplotlib import font_manager,rc
font_path='/Library/Fonts/Arial Unicode.ttf'
font=font_manager.FontProperties(fname=font_path).get_name()
rc('font',family=font)
X=df[["공학여부","weight","만나이","수면시간"]]
y=df["적정여부_rev"]

X=pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.2)

model_lr=LogisticRegression()
model_lr.fit(X_train,y_train)

model_xgb=XGBClassifier()
model_xgb.fit(X_train,y_train)

np.exp(model_lr.coef_)

plt.bar(height=model_xgb.feature_importances_,x=model_xgb.feature_names_in_)
plt.show()
```



In [42]:
```python
# 4-4. 두 모델의 roc-auc 그래프를 하나의 그래프에 겹쳐 그려라
from sklearn.metrics import roc_curve

fig,ax=plt.subplots()
from sklearn.metrics import roc_curve,roc_auc_score
fpr,tpr,thres= roc_curve(y_test,model_lr.predict_proba(X_test)[:,1],pos_label=1)
plt.plot(fpr,tpr,color='b')
fpr,tpr,thres= roc_curve(y_test,model_xgb.predict_proba(X_test)[:,1],pos_label=1)
plt.plot(fpr,tpr,color='orange')
# print(roc_auc_score(y_test,model_xgb.predict_proba(X_test)[:,1]))
plt.legend(['lr','xgb'])
plt.show()
```



In [43]:
```
# 5. 평균이 θ이고 분산이 100인 정규분포의 사전분포가 100, 256일때 120의 값을 가지는 데이터가 있을 경우 사후평균은?
# 좋아요! 이건 전형적인 정규–정규 공액 사전/사후 문제입니다. 차근차근 계산해볼게요.
# 1. 문제 해석

# 데이터
# x=120 (표본 1개)

# 데이터 분포:

# X~N(θ,σ2), σ2=100
# 사전분포: θ~N(μ0,τ02)

# 여기서 사전평균(μ₀)은 100, 분산(τ₀²)은 256 (표준편차 = 16) n=1 임.
```

2. 사후분포 공식 (정규-정규 공액)

사후분포:

$$\theta \mid x \sim N(\mu_n, \tau_n^2)$$

여기서

- **사후분산**

$$\tau_n^2 = \left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2} \right)^{-1}$$

- **사후평균**

$$\mu_n = \tau_n^2 \left(\frac{\mu_0}{\tau_0^2} + \frac{n\bar{x}}{\sigma^2} \right)$$

여기서 $n = 1, \bar{x} = x$

---

3. 계산 단계

(1) 주어진 값

- $\mu_0 = 100,\ \tau_0^2 = 256,\ \sigma^2 = 100,\ x = 120$

(2) precision (precision = 1/variance)

- prior precision = $1/\tau_0^2 = 1/256 = 0.00390625$
- likelihood precision = $1/\sigma^2 = 1/100 = 0.01$
- posterior precision = $0.00390625 + 0.01 = 0.01390625$

따라서 posterior variance:

$$\tau_n^2 = 1/0.01390625 = 71.9$$

(3) posterior mean

$$\mu_n = 71.9 \left(\frac{100}{256} + \frac{120}{100} \right) = 71.9 \left(0.390625 + 1.2 \right) = 71.9 \times 1.590625 \approx 114.4$$

---

4. 직관적 해석 (가중 평균 형태)

사후평균은 사실 사전평균과 표본평균의 precision-weighted average:

$$\mu_n = \frac{\frac{1}{\tau_0^2}\mu_0 + \frac{1}{\sigma^2}x}{\frac{1}{\tau_0^2} + \frac{1}{\sigma^2}} = \frac{0.00390625 \cdot 100 + 0.01 \cdot 120}{0.01390625} = \frac{0.390625 + 1.2}{0.01390625} \approx 114.4$$

즉, prior=100, data=120 → 사후평균은 그 중간값 쯤으로, 표본 쪽이 조금 더 무게가 큼 (likelihood precision > prior precision).

✅ 정답

사후평균 ≈ 114.4

In [44]:
```
# 6. 데이터 설명 : TV, Radio, Newspaper에 각각 광고비(달러)를 다르게 했을때 매출액 (Sales , 밀리언달러)를 나타내는 데이터(종속변수 : Sales)
df_6 =pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_5_advertising.csv')
df_6.head()
```

Out[44]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

In [45]:
```
# 6-1 회귀 모델링 후 유의하지 않는변수 파악
from statsmodels.formula.api import ols

model_ols_6=ols(data=df_6,formula='Sales~TV+Radio+Newspaper').fit()
model_ols_6.summary()
```

Out[45]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.903
Model:	OLS	**Adj. R-squared:**	0.901
Method:	Least Squares	**F-statistic:**	605.4
Date:	Thu, 09 Oct 2025	**Prob (F-statistic):**	8.13e-99
Time:	11:43:47	**Log-Likelihood:**	-383.34
No. Observations:	200	**AIC:**	774.7
Df Residuals:	196	**BIC:**	787.9
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.6251	0.308	15.041	0.000	4.019	5.232
TV	0.0544	0.001	39.592	0.000	0.052	0.057
Radio	0.1070	0.008	12.604	0.000	0.090	0.124
Newspaper	0.0003	0.006	0.058	0.954	-0.011	0.012

Omnibus:	16.081	**Durbin-Watson:**	2.251
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	27.655
Skew:	-0.431	**Prob(JB):**	9.88e-07
Kurtosis:	4.605	**Cond. No.**	454.

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

In [46]:
```
# 6-2 변수 선택시 먼저 제거 될 변수 및 근거 제시
#  Newspaper pvalue(0.954) > 0.05로 변수 선택시 먼저 제거되어야 함
```

In [47]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
vif=[variance_inflation_factor(df_6[['TV', 'Radio','Newspaper']].values,i) for i in range(3)]

pd.DataFrame(index=['TV', 'Radio','Newspaper'],columns=["VIF"],data=vif)
# VIF<10으로 변수 간 다중공선성에 대한 영향 없다고 판단함
```

Out[47]:

	VIF
TV	2.486772
Radio	3.285462
Newspaper	3.055245

In [48]:
```
# 7. 데이터 설명 : A,B,C,D,E 영업사원의 각 계약 성사 유무 (1:계약 , 0:미계약) 를 나타낸 데이터이다.

# 영업사원의 평균 계약 성사 건수는 같은지 통계 검정하라
df_7 =pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_7.csv',index_col = 0)
df_7.head()
```

Out[48]:

	A	B	C	D	E
contract_1	1	0	1	1	1
contract_2	0	1	0	1	0
contract_3	1	0	0	0	0
contract_4	0	1	1	1	1
contract_5	0	1	0	1	0

In [49]:
```
from scipy.stats import chi2_contingency
chi2_contingency(df_7)
```

Out[49]:
```
Chi2ContingencyResult(statistic=27.142857142857142, pvalue=0.9784871722517752, dof=44, expected_freq=array([[0.93333333, 1.066
66667, 0.53333333, 0.93333333, 0.53333333],
       [0.46666667, 0.53333333, 0.26666667, 0.46666667, 0.26666667],
       [0.23333333, 0.26666667, 0.13333333, 0.23333333, 0.13333333],
       [0.93333333, 1.06666667, 0.53333333, 0.93333333, 0.53333333],
       [0.46666667, 0.53333333, 0.26666667, 0.46666667, 0.26666667],
       [0.93333333, 1.06666667, 0.53333333, 0.93333333, 0.53333333],
       [0.46666667, 0.53333333, 0.26666667, 0.46666667, 0.26666667],
       [0.23333333, 0.26666667, 0.13333333, 0.23333333, 0.13333333],
       [0.7       , 0.8       , 0.4       , 0.7       , 0.4       ],
       [0.46666667, 0.53333333, 0.26666667, 0.46666667, 0.26666667],
       [0.46666667, 0.53333333, 0.26666667, 0.46666667, 0.26666667],
       [0.7       , 0.8       , 0.4       , 0.7       , 0.4       ]]))
```

In [50]:
```python
# 8-1. 유기견이 하루에 2.2마리 발생한다.한마리도 안 버려질 확률을 구하라
# 8-2. 적어도 2마리가 버려질 확률을 구하라

import math
from scipy.stats import poisson

lam = 2.2

# 8-1
p0 = poisson.pmf(0, lam)
print("P(X=0) =", p0)

# 8-2
p_ge2 = 1 - poisson.cdf(1, lam)
print("P(X>=2) =", p_ge2)
```

```
P(X=0) = 0.11080315836233387
P(X>=2) = 0.6454298932405317
```

In [ ]: