

t3_p1_prac

검정 종류	직접 출제 비중	출제 방식
단일표본 t-검정	매우 드뭄	명시적 출제 거의 없음
독립/대응표본 t	자주 나옴	예: 두 집단 전후 비교 등
카이제곱 독립성	매우 자주 나옴	범주형 변수 간 연관성 판단
신뢰구간 계산	종종 나옴	stats.t.interval() 형태로 출제
statsmodels 회귀	자주 나옴	계수 유의성, p-value 등 분석

단일 표본 t 검정 (단측)

문제: 다음은 22명의 학생들이 국어시험에서 받은 점수이다. 학생들의 평균이 75보다 크다고 할 수 있는가?

- 귀무가설(H0): 모평균은 mu와 같다. ($\mu = \mu_0$), 학생들의 평균은 75이다
- 대립가설(H1): 모평균은 mu보다 크다. ($\mu > \mu_0$), 학생들의 평균은 75보다 크다

가정:

- 모집단은 정규분포를 따른다.
- 표본의 크기가 충분히 크다.

검정통계량, p-value, 검정결과를 출력하시오

```
from scipy import stats
import pandas as pd
import numpy as np

# 데이터
scores = [75, 80, 68, 72, 77, 82, 81, 79, 70, 74, 76, 78, 81, 73, 81, 78, 75, 72, 74, 79, 78, 79]

mu= 75
sample_mean = np.mean(scores)
sample_mean

76.45454545454545

stats.ttest_1samp(scores,mu, alternative = 'greater')

TtestResult(statistic=1.765879233231226, pvalue=0.04597614747709146, df=21)

# pvalue <0.05 귀무가설 탈락, 모평균은 75보다 크다고 할 수 있다.

# 검정통계량
# t 통계량은 ***표본평균과 모평균이 얼마나 떨어져 있는지를 표준오차 단위로 나타낸 값***입니다.
```

```
# help(ttest_1samp)
```

```
# import scipy.stats
# dir(scipy.stats)
```

- 기본 형태
t_stat, p_val = ttest_1samp(data, popmean, alternative='two-sided')

data: 샘플 데이터 (리스트 또는 배열)

popmean: 비교할 모평균 (예: 75)

alternative: 대립가설의 방향 설정

'two-sided' (기본값): $\mu \neq \text{popmean}$

'greater': $\mu > \text{popmean}$

'less': $\mu < \text{popmean}$

유형	비교 대상	사용 조건 (주요 예시)
단일표본	표본 평균 vs 모평균	“이 표본 평균이 $\mu = 75$ 보다 크다?”
독립표본	두 개의 독립된 집단 평균	“복용군 vs 비복용군의 평균 체온 차이?”
대응표본	같은 집단의 전/후 평균	“치료 전후 체중 변화가 유의한가?”

독립표본 t 검정

문제 : 어떤 특정 약물을 복용한 사람들의 평균 체온이 복용하지 않은 사람들의 평균 체온과 유의미하게 다른지 검정해보려고 합니다.

가정:

약물을 복용한 그룹과 복용하지 않은 그룹의 체온 데이터가 각각 주어져 있다고 가정합니다.
각 그룹의 체온은 정규분포를 따른다고 가정합니다.

검정통계량, p-value, 검정결과를 출력하시오

```
from scipy import stats

# 가설 설정
# H0 : 약물을 복용한 그룹과 복용하지 않은 그룹의 평균 체온은 유의미한 차이가 없다.
# H1 : 약물을 복용한 그룹과 복용하지 않은 그룹의 평균 체온은 유의미한 차이가 있다.
```

```
# 데이터 수집
group1 = [36.8, 36.7, 37.1, 36.9, 37.2, 36.8, 36.9, 37.1, 36.7, 37.1]
group2 = [36.5, 36.6, 36.3, 36.6, 36.9, 36.7, 36.7, 36.8, 36.5, 36.7]
```

```
stats.ttest_ind(group1,group2) # two-sided
```

```
TtestResult(statistic=3.7964208654863336, pvalue=0.001321891476703691, df=18.0)
```

```
stats.levene(group1,group2) # 등분산 가정을 하지않고 직접 검증>>등분산
```

```
LeveneResult(statistic=0.19889502762431177, pvalue=0.6609323607193374)
```

```
stats.ttest_ind(group1,group2,equal_var=True) # two-sided equal_var=True 명시. 기본값.
```

```
TtestResult(statistic=3.7964208654863336, pvalue=0.001321891476703691, df=18.0)
```

상황	사용 함수	설명
등분산 가정됨	ttest_ind(..., equal_var=True)	고전적 독립표본 t-검정
등분산 가정 안됨	ttest_ind(..., equal_var=False)	Welch's t-검정, 현실에서 더 안전

```
# help(stats.ttest_ind)
```

대응 표본 t 검정

문제 : 주어진 데이터는 고혈압 환자 치료 전후의 혈압이다. 해당 치료가 효과가 있는지 대응(쌍체)표본 t-검정을 진행하시오

- 귀무가설(H0): $\mu \geq 0$
- 대립가설(H1): $\mu < 0$
- μ = (치료 후 혈압 - 치료 전 혈압)의 평균
- 유의수준: 0.05

- μ 의 표본평균은?(소수 둘째자리까지 반올림)
- 검정통계량 값은?(소수 넷째자리까지 반올림)
- p-값은?(소수 넷째자리까지 반올림)
- 가설검정의 결과는? (유의수준 5%)

```
import pandas as pd
df=pd.read_csv('data/high_blood_pressure.csv')
```

```
df.head(2)
```

	Id	sex	age	bp_pre	bp_post
0	p001	Male	33	149	129
1	p002	Male	39	168	168

```
stats.ttest_rel(df['bp_post'],df['bp_pre'],alternative = 'less')
```

```
TtestResult(statistic=-3.0002163948827545, pvalue=0.0016434474228511028, df=119)
```

일원분산분석 (ANOVA)

문제 : 세 가지 다른 교육 방법(A, B, C)을 사용하여 수험생들의 시험 성적을 개선시키는 효과를 평가하고자 한다.

30명의 학생들을 무작위로 세 그룹으로 배정하여 교육을 실시하였고, 시험을 보고 성적을 측정하였습니다. 다음은 각 그룹의 학생들의 성적 데이터입니다.

귀무가설(H_0): 세 그룹(A, B, C) 간의 평균 성적 차이가 없다.
대립가설(H_1 또는 H_a): 세 그룹(A, B, C) 간의 평균 성적 차이가 있다.

일원분산분석(One-Way ANOVA)는 말 그대로 “3개 이상의 집단 평균을 비교하는 확장된 t-검정”

```
# 각 그룹의 데이터
groupA = [85, 92, 78, 88, 83, 90, 76, 84, 92, 87]
groupB = [79, 69, 84, 78, 79, 83, 79, 81, 86, 88]
groupC = [75, 68, 74, 65, 77, 72, 70, 73, 78, 75]
```

-다음 주어진 데이터로 일원배치법을 수행하여 그룹 간의 평균 성적 차이가 있는지 검정하세요

- f값 (소수 둘째자리)
- p값 (소수 여섯째자리)
- 검정결과 출력

```
# dir(scipy.stats)
```

```
from scipy.stats import f_oneway
```

```
# help(f_oneway)
```

```
stats,p_val= f_oneway(groupA,groupB,groupC)
```

```
round(stats,2)
```

16.88

```
round(p_val,6)
```

1.8e-05

```
import pandas as pd

groupA = [85, 92, 78, 88, 83, 90, 76, 84, 92, 87]
groupB = [79, 69, 84, 78, 79, 83, 79, 81, 86, 88]
groupC = [75, 68, 74, 65, 77, 72, 70, 73, 78, 75]

data = {'GroupA': groupA, 'GroupB': groupB, 'GroupC': groupC}
```

```
df_wide = pd.DataFrame(data)
```

```
df_wide
```

	GroupA	GroupB	GroupC
0	85	79	75
1	92	69	68
2	78	84	74
3	88	78	65
4	83	79	77
5	90	83	72
6	76	79	70
7	84	81	73
8	92	86	78
9	87	88	75

```
df_long = df_wide.melt(value_vars=['GroupA', 'GroupB', 'GroupC'], var_name='Group', value_name='Score')
```

```
# df_long
```

```
# help(df.melt)
```

```
df_pivot = df_long.reset_index().pivot(index='index', columns='Group', values='Score')
```

```
# df_pivot
```

정규성 검정 (Shapiro-Wilk)

문제 : 12명의 수험생이 빅데이터 분석기사 시험에서 받은 점수이다. Shapiro-Wilk 검정을 사용하여 데이터가 정규 분포를 따르는지 검증하시오

귀무 가설(H_0): 데이터는 정규 분포를 따른다.

대립 가설(H_1): 데이터는 정규 분포를 따르지 않는다.

Shapiro-Wilk 검정 통계량, p-value, 검증결과를 출력하시오

```
# dir(scipy.stats)
from scipy.stats import shapiro
# help(shapiro)
```

```
data = [75, 83, 81, 92, 68, 77, 78, 80, 85, 95, 79, 89]
```

```
statistic, p_val = shapiro(data)
statistic, p_val
```

(0.9768091723993144, 0.9676506711851194)

귀무가설 기각 실패. 정규 분포를 따른다.

회귀모형 (상관계수)

문제 : iris 데이터에서 Sepal Length와 Sepal Width의 상관계수 계산하고 소수 둘째자리까지 출력하시오

```
import pandas as pd
from sklearn.datasets import load_iris

# iris 데이터셋 로드
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
```

```
cor=df.corr()
```

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
cor.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	1.000000	-0.117570	0.871754	0.817941
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126
petal length (cm)	0.871754	-0.428440	1.000000	0.962865
petal width (cm)	0.817941	-0.366126	0.962865	1.000000

```
round(cor.iloc[0,1],2)
```

-0.12

로지스틱 회귀

문제 : 타이타닉 데이터에서

Pclass, Gender, sibsp, parch를 독립변수로 사용하여 로지스틱 회귀모형을 실시하였을 때, parch변수의 계수값은? 단, Pclass는 범주형 변수이다

(반올림하여 소수 셋째 자리까지 계산)

```
import pandas as pd
df= pd.read_csv('data/Titanic.csv')
```

```
from statsmodels.formula.api import logit
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   PassengerId     891 non-null    int64
 1   Survived        891 non-null    int64
 2   Pclass          891 non-null    int64
 3   Name            891 non-null    object
 4   Gender          891 non-null    object
 5   Age             714 non-null    float64
 6   SibSp           891 non-null    int64
 7   Parch           891 non-null    int64
 8   Ticket          891 non-null    object
 9   Fare            891 non-null    float64
10   Cabin           204 non-null    object
11   Embarked        889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
for col in ['Pclass','Gender','SibSp','Parch','Survived']:
    print(df[col].value_counts())
```

```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
Gender
male      577
female    314
Name: count, dtype: int64
SibSp
0     608
1     209
2       28
4       18
3       16
8        7
5         5
Name: count, dtype: int64
Parch
0     678
1     118
```

```
2      80
5       5
3       5
4       4
6       1
Name: count, dtype: int64
Survived
0      549
1      342
Name: count, dtype: int64
```

```
formula = 'Survived ~ C(Pclass) + Gender + SibSp + Parch'

model = logit(formula, df).fit()
model.summary()
model.params
```

```
Optimization terminated successfully.
      Current function value: 0.459565
      Iterations 6
```

```
Intercept          2.491729
C(Pclass)[T.2]     -0.848152
C(Pclass)[T.3]     -1.866905
Gender[T.male]     -2.760281
SibSp              -0.232553
Parch              -0.049847
dtype: float64
```

```
logP= model.params['Parch']
logP
```

```
-0.04984725834216791
```

```
np.exp(logP)
# Parch가 1 증가할 때, 생존할 **오즈(odds)**는 약 0.95배로 감소합니다.
# 즉, Parch가 많을수록 생존 확률이 약간 낮아지는 경향이 있음 (다른 변수들이 동일할 때)
# 생존확률 p = odds / (1+odds)
```

```
0.9513747279566538
```

```
# C()는 독립변수가 숫자일 때, 범주형으로 처리하라는 명령어
# 종속변수는 모델 종류에 따라 자동으로 적절히 처리됨
```

✓ 로지스틱 회귀 해석 핵심

****계수(β)****는 log-odds에 대한 영향력

$\exp(\beta)\exp(\beta) \rightarrow$ 오즈비(Odds Ratio) 해석 가능

예: parch 계수가 0.2면, parch가 1 증가할 때 생존할 log-odds가 0.2 증가

Pclass는 범주형이므로 \rightarrow C(Pclass) 처리

Gender는 문자열이므로 \rightarrow df['Gender'].map({'male':0, 'female':1}) 등 변환

logit()으로 적합

params['parch']의 값을 소수 셋째 자리까지 반올림

항목	설명
종속변수	Survived (0/1 이진값)
독립변수	Pclass (범주형), Gender (이진으로 인코딩), sibsp, parch
회귀모형	로지스틱 회귀 (Logit 함수 사용)
회귀계수	로그 오즈비 (log-odds)
영향력 해석	$\exp(\beta) \rightarrow$ 오즈비(Odds Ratio) 로 해석 가능

요소	설명
~	왼쪽은 종속변수(Y), 오른쪽은 독립변수(X들)
+	설명변수 간 단순 추가(선형결합)
C(변수명)	범주형(categorical) 변수임을 명시
숫자형 변수	그냥 이름만 쓰면 됨 (예: Age, Parch)
자동 더미화	c() 를 쓰면 자동으로 원-핫 인코딩 처리됨

statsmodels.formula.api는 통계모델을 R 스타일의 formula 문자열로 표현할 수 있게 해주는 매우 실용적인 인터페이스입니다.

함수명	설명	예시
ols()	선형 회귀(Ordinary Least Squares)	"Y ~ X1 + X2"
logit()	로지스틱 회귀 (이항 분류)	"Y ~ X1 + C(X2)"
glm()	일반화 선형모형 (Poisson, Binomial 등)	"Y ~ X1" + family=Poisson()
mnlogit()	다항 로지스틱 회귀 (3개 이상 클래스)	"Y ~ X1 + X2"
mixedlm()	혼합효과모형 (랜덤효과 포함)	"Y ~ X1"
r1m()	강건 회귀 (이상치에 강한)	"Y ~ X1"
gee()	일반화 추정 방정식 (반복측정)	"Y ~ X1"

- ols() — 선형 회귀

```
from statsmodels.formula.api import ols
model = ols("score ~ age + hours+ C(Sex)", data=df).fit()
```

- logit() — 로지스틱 회귀

```
from statsmodels.formula.api import logit
model = logit("Survived ~ C(Sex) + Age", data=df).fit()
```

두 그룹 평균비교 (T/F 검정)

문제: 두 교육 방법의 효과 비교

연구자는 두 가지 다른 교육 방법이 학생들의 성적에 미치는 영향을 비교하고자 합니다. 연구자는 무작위로 선발된 20명의 학생들을 두 그룹으로 나누어 한 그룹에는 교육 방법 A를, 다른 그룹에는 교육 방법 B를 적용합니다. 교육이 끝난 후, 두 그룹의 성적을 비교하기 위해 독립 표본 t-검정과 ANOVA F-검정을 실시하려고 합니다.

다음은 두 그룹의 성적입니다: 다음의 두 가지 검정을 사용하여 두 교육 방법 간의 성적 차이가 통계적으로 유의한지를 검증하세요

독립 표본 t-검정을 실시하여 t 통계량을 구하세요.

독립 표본 t-검정을 실시하여 p-값을 구하세요.

ANOVA F-검정을 실시하여 F 통계량을 구하세요.

ANOVA F-검정을 실시하여 p-값을 구하세요.

```
import pandas as pd
df = pd.DataFrame({
    'A':[77, 75, 82, 80, 81, 83, 84, 76, 75, 87],
    'B':[80, 74, 77, 79, 71, 74, 78, 69, 70, 72],
})
```

```
from scipy import stats
```

```
t_stats, p_val = stats.ttest_ind(df['A'],df['B']) # 기본 two-sided 사용
t_anova, p_anova = stats.f_oneway(df['A'],df['B'])
```

```
print(f"독립표본 t검정 : {t_stats}, {p_val}")
```

독립표본 t검정 : 3.1068522301122954, 0.006087373605949963

```
#
```

```
print(f"ANOVA t검정 : {t_anova}, {p_anova}")
```

ANOVA t검정 : 9.652530779753763, 0.006087373605949924

- 2그룹일 때: ttest_ind()와 f_oneway() → 동등한 결론
- 3그룹 이상일 때: ttest_ind()는 불가능, → ANOVA만 가능
- p-value는 동일, 검정통계량만 다름 ($F = t^2$)

적합도 검정 (Chi-square)

문제 : 카이제곱 적합도 검정

고등학교에서는 졸업생들이 선택하는 대학 전공 분야의 선호도가 시간이 지남에 따라 변하지 않는다고 가정합니다. 학교 측은 최근 졸업생들의 전공 선택이 과거와 같은 패턴을 따르는지 알아보기 위해 적합도 검정을 실시하기로 결정했습니다.

과거 자료에 따르면 졸업생들이 선택하는 전공의 분포는 다음과 같습니다:

인문학: 20% 사회과학: 30% 자연과학: 25% 공학: 15% 기타: 10% 올해 졸업한 학생 200명의 전공 선택 분포는 다음과 같았습니다:

인문학: 30명 사회과학: 60명 자연과학: 50명 공학: 40명 기타: 20명 이 데이터를 바탕으로, 졸업생들의 전공 선택 패턴이 과거와 유사한지를 알아보기 위해 카이제곱 적합도 검정을 실시해야 합니다. 유의 수준은 0.05로 설정합니다.

검정 통계량?
p-value?
유의수준 하 귀무가설 기각 또는 채택?

```
# 관측된 빈도
observed_frequencies = [30, 60, 50, 40, 20]
# 기대된 빈도
expected_frequencies = [200 * 0.20, 200 * 0.30, 200 * 0.25, 200 * 0.15, 200 * 0.10]
```

```
from scipy.stats import chisquare
```

```
chisquare(observed_frequencies, expected_frequencies)
# chisquare(f_obs=observed_frequencies, f_exp=expected_frequencies)
```

```
Power_divergenceResult(statistic=5.833333333333334, pvalue=0.21194558437271782)
```

```
# 귀무가설 : 올해 졸업생들의 전공 선택 분포는 과거와 동일하다
# 대립가설 : 올해 졸업생들의 전공 선택 분포는 과거와 다르다
```

```
# ✅ 보너스: 자유도 (df)

# 자유도는:
# df=k-1=5-1=4
# df=k-1=5-1=4

# → 이 정보는 시험에서 검정표나 임계값으로 기각여부를 따질 때 참고할 수 있어요.
```

카이제곱 적합도 검정과 독립표본 t-검정은 모두 두 그룹 이상의 차이를 비교하긴 하지만, 비교하는 대상과 데이터의 성격이 완전히 다릅니다.

구분	독립표본 t-검정	카이제곱 적합도 검정
목적	두 집단의 평균 차이 비교	관측값과 기대값의 빈도 차이 비교
데이터	연속형 데이터 (ex. 점수)	범주형 데이터의 빈도
가설 예시	교육방법 A vs B의 평균 점수 차이	전공 비율이 과거와 같은가
통계량	t-통계량	χ^2 (카이제곱) 통계량
함수	scipy.stats.ttest_ind()	scipy.stats.chisquare()
사용 조건	그룹마다 연속형 데이터 샘플이 있어야 함	관측 빈도와 기대 비율만 있으면 가능

scipy.stats.chi2_contingency()는 카이제곱 독립성 검정

- 두 개의 범주형 변수 간에 독립적인지, 관련이 있는지를 검정할 때 사용합니다.

예시	설명
성별 vs 흡연 여부	남/여와 흡연/비흡연의 관계가 있는가?
지역 vs 선호 음식	지역별로 음식 선호가 다른가?
전공 vs 취업여부	전공과 취업을 간에 연관이 있는가?

ttest_ind는 평균값의 차이를 보고, chisquare는 범주(분류)의 비율(빈도) 차이를 본다.

지지도/신뢰도/향상도

문제 : 편의점 물건이 팔린 예

빼빼로'와 '딴짓초코'가 함께 팔린 거래의 지지도를 계산하세요.

'빼빼로'가 팔린 거래 중에서 '빼빼로'와 '오징어칩'이 함께 팔린 거래의 신뢰도를 계산하세요.

'빼빼로'와 '양조위빵'의 향상도를 계산하세요.

지지도(A,B): A와 B가 함께 팔린 거래 횟수 / 전체 거래 횟수
신뢰도(A->B): A와 B가 함께 팔린 거래 횟수 / A가 팔린 거래 횟수
향상도(A,B): 신뢰도(A->B) / 지지도(B)

```
import pandas as pd
# 데이터
df = pd.DataFrame({
    'transaction': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    '빼빼로': [1, 0, 1, 1, 0, 1, 1, 0, 1, 1],
    '딴짓초코': [0, 1, 1, 0, 1, 0, 1, 1, 0, 0],
    '양조위빵': [1, 0, 0, 1, 1, 1, 0, 0, 1, 0],
    '오징어칩': [0, 1, 1, 0, 0, 1, 0, 1, 1, 1],
    '초코파이': [1, 1, 0, 0, 1, 0, 1, 1, 0, 0]
})
df
```

	transaction	빼빼로	딴짓초코	양조위빵	오징어칩	초코파이
0	1	1	0	1	0	1
1	2	0	1	0	1	1
2	3	1	1	0	1	0
3	4	1	0	1	0	0
4	5	0	1	1	0	1
5	6	1	0	1	1	0
6	7	1	1	0	0	1
7	8	0	1	0	1	1
8	9	1	0	1	1	0
9	10	1	0	0	1	0

```
# 각 문제의 계산을 위한 데이터 준비
total = df.shape[0]
pepero = df['빼빼로'].sum()
```

```
# 문제 1: 빼빼로와 딴짓초코가 함께 팔린 거래의 지지도
pepero_and_choco = len(df[(df['빼빼로'] == 1) & (df['딴짓초코'] == 1)])
print(pepero_and_choco / total)
```

0.2

```
# 문제 2: 뽀빠리가 팔린 거래 중 뽀빠리와 오징어칩이 함께 팔린 거래의 신뢰도
pepero_and_squid = len(df[(df['뽀빠리'] == 1) & (df['오징어칩'] == 1)])
print(pepero_and_squid / pepero)
```

0.5714285714285714

```
# 문제 3: 뽀빠리와 양조위빵의 향상도
pepero_and_bread = len(df[(df['뽀빠리'] == 1) & (df['양조위빵'] == 1)])
bread = df['양조위빵'].sum()
print((pepero_and_bread / pepero) / (bread / total))
```

1.1428571428571428

bread

5

pepero_and_bread

4

개념	수식	분모 기준	해석
지지도(A)	$지지도(A) = P(A) = A발생비율$	전체 거래 수	A가 얼마나 자주 등장하는가
지지도(A ∩ B)	$P(A \cap B)$	전체 거래 수	A와 B가 동시에 발생하는 비율
신뢰도(A → B)	$P(B A) = P(A \cap B) / P(A)$	A 발생 수	A가 발생했을 때 B도 발생할 확률
향상도(A → B)	$신뢰도(A \rightarrow B) / 지지도(B)$	-	A가 발생하면 B가 더 많이 일어나는지 (1보다 크면 양의 연관)

포아송 분포

문제: 한 서점에서는 평균적으로 하루에 3명의 고객이 특정 잡지를 구매합니다. 이 데이터는 포아송 분포를 따른다고 가정할 때, 다음 질문에 대한 답을 구하세요.

하루에 정확히 5명의 고객이 잡지를 구매할 확률은 얼마입니까? (%로 값을 정수로 입력하십시오)
하루에 적어도 2명의 고객이 잡지를 구매할 확률은 얼마입니까? (%로 값을 정수로 입력하십시오)

```
from scipy.stats import poisson
```

```
# help(poisson)
# pmf(k, mu, loc=0)
# Probability mass function.
# cdf(k, mu, loc=0)
# Cumulative distribution function.
```

```
from scipy.stats import poisson
```

```
# 평균 발생 횟수 (하루에 잡지를 구매하는 고객 수)  
lambda_ = 3
```

```
# 하루에 정확히 5명의 고객이 잡지를 구매할 확률  
print(poisson.pmf(5, lambda_))
```

```
0.10081881344492458
```

```
# 10%
```

```
# 하루에 적어도 2명의 고객이 잡지를 구매할 확률  
print(1 - poisson.cdf(1, lambda_))
```

```
0.8008517265285442
```

함수	역할	설명
pmf(k, mu)	확률 질량 함수	정확히 k번 발생할 확률
cdf(k, mu)	누적 분포 함수	k번 이하로 발생할 확률 ($P(X \leq k)$)
sf(k, mu)	상위 누적	k번 초과 확률 ($P(X > k) = 1 - \text{cdf}(k, \text{mu})$)

독립성 검정 (chi2_contingency)

문제 : 성별과 시험합격은 독립적인가를 검정하시오!

- 검정 통계량?
- p-value?
- 귀무가설 기준 (기각/채택)?
- 남자의 합격 기대 빈도?

```
male = [100, 200]  
female= [130, 170]  
  
data= {"남자": male, "여자":female}  
df=pd.DataFrame(data, index= ['합격', '불합격'])
```

```
df
```

	남자	여자
합격	100	130
불합격	200	170

```
statistic, pvalue, dof, expected= stats.chi2_contingency(df)
```

```
print (statistic, pvalue, dof,expected)
```

```
5.929494712103407 0.01488951060599475 1 [[115. 115.]
[185. 185.]]
```

```
# 결과 출력
print(f'검정통계량: {statistic}')
print(f'p-value: {pvalue}')
print(f'남자의 합격 기대빈도: {expected[0][0]}')
```

```
검정통계량: 5.929494712103407
p-value: 0.01488951060599475
남자의 합격 기대빈도: 115.0
```

```
stats.chi2_contingency(df)
```

```
Chi2ContingencyResult(statistic=5.929494712103407, pvalue=0.01488951060599475, dof=1, expected_freq=array([[115.,
115.],
[185., 185.])))
```

```
# 유의수준 5%에서 성별과 시험 합격 간에는 통계적으로 유의한 관계가 있으며, 두 변수는 독립적이지 않다.
# 특히, 남자는 기대보다 적게 합격했고, 여자는 기대보다 많이 합격하였다.
# [[115. 115.], [185. 185.]]는 chi2_contingency() 함수의 반환값 중 **기대빈도표 (expected)**이고,그 안에 있는 185는 불합
격자의 기대값
```

베르누이 분포와 이항분포

문제 : 베르누이 분포와 이항분포

```
[베르누이 분포] 다음 데이터는 100번의 시도에서 각각 성공(1) 또는 실패(0)를 나타냅니다. 이 데이터를 바탕으로 각 시도의 성공 확
률을 계산하시오.
[이항분포] 1번 문제에서 계산한 성공 확률을 사용하여, 100번의 시도 중 정확히 60번 성공할 확률을 계산하시오.
```

```
df= pd.read_csv('data/type3/t3_success.csv')
```

```
df
```

	Success
0	1
1	0
2	0
3	1
4	1
...	...
95	1
96	1

	Success
97	1
98	0
99	1

100 rows × 1 columns

```
from scipy.stats import binom
```

```
# 베르누이 분포 : 각 시도의 성공 확률 계산
total_attempts = len(df)
number_of_successes = df['Success'].sum()
success_probability = number_of_successes / total_attempts
```

```
success_probability
```

```
0.62
```

```
# 이항분포 : 100 번의 시도 중 정확히 60번 성공할 확률 계산
n=100 # 시도 횟수
k=60 # 성공 횟수
prob_60_success= binom.pmf(k,n,success_probability)
```

```
print(prob_60_success)
```

```
0.07464985555860272
```

```
# help(binom)
# pmf(k, n, p, loc=0)
#         Probability mass function.
# cdf(k, n, p, loc=0)
#         Cumulative distribution function.
```

```
# 60번 이상 성공할 확률 계산
# over_60= 1-binom.cdf(k,n,success_probability)
over_60 = 1 - binom.cdf(59, n=100, p=0.6)
```

```
print(over_60)
```

```
0.54329448588207
```

```
help(binom.cdf)
```

```
Help on method cdf in module scipy.stats._distn_infrastructure:
```



```
cdf(k, *args, **kwargs) method of scipy.stats._discrete_distns.binom_gen instance
Cumulative distribution function of the given RV.

Parameters
-----
k : array_like, int
    Quantiles.
arg1, arg2, arg3,... : array_like
    The shape parameter(s) for the distribution (see docstring of the
    instance object for more information).
loc : array_like, optional
    Location parameter (default=0).

Returns
-----
cdf : ndarray
    Cumulative distribution function evaluated at `k`.
```

문제: 확률이 0.7인 시험을 10번 응시했을 때,

8번 이상 합격할 확률은?

```
from scipy.stats import binom

prob = 1 - binom.cdf(7, n=10, p=0.7)
print(f"P(X ≥ 8) = {round(prob, 4)}")
```

P(X ≥ 8) = 0.3828

```
# 정확히 60번 성공할 확률은?
binom.pmf(60, n=100, p=0.6)
```

0.08121914499610604

분포	설명	핵심 파라미터	예시
베르누이(Bernoulli)	1번 시도에서 성공(1)/실패(0)	성공확률 p	동전 1번 던지기
이항(Binomial)	n번 반복된 베르누이 시행 중 성공 횟수	n, p	동전 10번 던져 3번 앞면
포아송(Poisson)	단위 시간/공간당 이벤트 발생 횟수	평균 발생횟수 λ (mu)	1시간에 평균 3건 전화, 5건 올 확률?

- 포아송 : `scipy.stats.poisson`, `poisson.pmf(5, lambda_)`
- 이항 : `scipy.stats.binom`, `binom.pmf(60, n=100, p=0.6)`
- 기억해야할 함수 `pmf`, `cdf`

점추정/구간추정 (t3-confidence_interval-py)

문제 : 데이터셋은 어떤 도시의 일일 평균 온도 입니다.

점추정: 데이터셋을 기반으로 이 도시의 평균 연간 온도를 점추정하세요. (반올림하여 소수 둘째자리까지)
구간추정: 95% 신뢰수준에서 이 도시의 평균 연간 온도에 대한 신뢰구간을 구하세요. (반올림하여 소수 둘째자리까지)

```
df=pd.read_csv('data/type3/daily_temperatures.csv')
```

```
df.head()
```

	Daily Average Temperature
0	28.820262
1	22.000786
2	24.893690
3	31.204466
4	29.337790

```
temperature_data=df.copy()
```

```
import pandas as pd
from scipy import stats
# CSV 파일 불러오기
# 점추정: 샘플 평균 계산
sample_mean = temperature_data['Daily Average Temperature'].mean()

# 구간추정: 샘플 표준편차 계산 및 신뢰구간 계산
confidence_level = 0.95
sample_std = temperature_data['Daily Average Temperature'].std() # 자유도 n-1로 설정 판다스는 자동으로 표본표준편차를
계산. 넘파이는 설정해야함. np.std(df,ddof=1)
n_samples = len(temperature_data)

# 95% 신뢰구간 계산
confidence_interval = stats.t.interval(confidence_level, df=n_samples-1, loc=sample_mean,
scale=sample_std/(n_samples**0.5))

sample_mean, confidence_interval
```

```
(19.937577543978538, (19.427887094620406, 20.44726799333667))
```

```
temp= df['Daily Average Temperature'].rename("온도")
```

```
temp
```

```
0      28.820262
1      22.000786
2      24.893690
3      31.204466
4      29.337790
...
360     23.492286
361     20.018854
362     24.659242
363     21.699825
364     19.921589
Name: 온도, Length: 365, dtype: float64
```

```
df = df.rename(columns={'Daily Average Temperature': '온도'})
```

```
std = df['온도'].std(ddof=1) # ddof=1 이 기본값이긴함
std
```

```
4.951746576285235
```

```
import numpy as np
import scipy.stats as stats

# 예시: df['온도']에 평균 온도 데이터가 있다고 가정
mean = df['온도'].mean()           # 점추정
std = df['온도'].std(ddof=1)        # 표본 표준편차
n = len(df['온도'])                # 표본 크기
se = std / np.sqrt(n)              # 표준오차

# 95% 신뢰구간 (t 분포 사용)
t_crit = stats.t.ppf(0.975, df=n-1) # 양쪽 2.5%씩 잘림

ci_lower = mean - t_crit * se
ci_upper = mean + t_crit * se

# 출력
print(f"점추정 (표본 평균): {round(mean, 2)}")
print(f"95% 신뢰구간: ({round(ci_lower, 2)}, {round(ci_upper, 2)})")
```

```
점추정 (표본 평균): 19.94
95% 신뢰구간: (19.43, 20.45)
```

매개변수	설명
confidence	신뢰수준 (예: 0.95 → 95% 신뢰구간)
df	자유도 (보통 <code>n - 1</code>)
loc	평균 (샘플 평균)
scale	표준오차 (standard error = 표준편차 / \sqrt{n})

```
# help(stats.t.interval)
```

```
import numpy as np
import scipy.stats as stats

# 평균, 표준편차, 표본 크기
mean = df['온도'].mean()
std = df['온도'].std(ddof=1)
n = len(df['온도'])
se = std / np.sqrt(n)

# 95% 신뢰구간 계산 (interval 함수 사용)
ci_lower, ci_upper = stats.t.interval(
    confidence=0.95,           # 신뢰수준
    df=n - 1,                  # 자유도
    loc=mean,                  # 중심 (표본 평균)
    scale=se                    # 표준오차
```

```
)

# 출력
print(f"점추정 (표본 평균): {round(mean, 2)}")
print(f"95% 신뢰구간: ({round(ci_lower, 2)}, {round(ci_upper, 2)})")
```

점추정 (표본 평균): 19.94
95% 신뢰구간: (19.43, 20.45)

```
help(stats.t.cdf)
```

```
Help on method cdf in module scipy.stats._distn_infrastructure:

cdf(x, *args, **kwargs) method of scipy.stats._continuous_distns.t_gen instance
    Cumulative distribution function of the given RV.

Parameters
-----
x : array_like
    quantiles
arg1, arg2, arg3,... : array_like
    The shape parameter(s) for the distribution (see docstring of the
    instance object for more information)
loc : array_like, optional
    location parameter (default=0)
scale : array_like, optional
    scale parameter (default=1)

Returns
-----
cdf : ndarray
    Cumulative distribution function evaluated at `x`
```

신뢰구간: (19.43°C, 20.45°C)
→ 95%의 신뢰 수준으로, 이 도시의 **모평균(진짜 평균 온도)**는 19.43도에서 20.45도 사이에 있을 것이라고 추정할 수 있습니다.

함수	역할	사용 목적
ppf()	임계값 계산	신뢰구간 ±t값
cdf()	누적확률	양측 검정 p-value
sf()	초과확률	단측 검정 p-value
interval()	신뢰구간 자동 계산	평균 ± t×표준오차

scipy.stats.t 는 **t-분포** 관련 함수들을 다루는 모듈이며, **통계 검정·신뢰구간 계산에 필수적인 함수들이** 들어 있습니다. 아래에 **자주 쓰는 함수 4개만 간단히** 정리해드릴게요.

✓ 1. ppf(q, df) : 퍼센트 포인트 함수

→ **t분포의 분위수**를 구함 (임계값 계산용)

```
from scipy.stats import t
t_crit = t.ppf(0.975, df=29) # 95% 신뢰구간 (양쪽 2.5% 제외)
```

✓ 2. `cdf(x, df)`: 누적 분포 함수

→ $P(T \leq x)$ 값을 계산 (p-value 구할 때 사용)

```
t.cdf(1.96, df=29)
```

`t.cdf(1.96, df)`에서 나오는 1.96은 **표준정규분포(또는 근사된 t-분포)**에서의 임계값을 의미합니다.
1.96의 의미

신뢰수준 95% 기준:

양쪽 2.5%씩 잘라낸 구간

가운데 95%를 포함하는 분포의 경계값

즉,

$P(-1.96 \leq Z \leq 1.96) = 0.95$

$P(-1.96 \leq Z \leq 1.96) = 0.95$

→ 그래서 양측 검정에서 유의수준 5%일 때,
임계값이 ± 1.96 이 됩니다 (표준정규 기준)

✓ 3. `sf(x, df)`: 상위 누적 분포

→ $P(T > x) = 1 - \text{cdf}(x)$ (단측 검정 p-value)

```
t.sf(1.96, df=29)
```

✓ 4. `interval(confidence, df, loc=mean, scale=se)`

→ **신뢰구간 계산**을 한 번에

```
t.interval(0.95, df=29, loc=19.94, scale=0.25)
```

✳ 요약표

함수	역할	사용 목적
<code>ppf()</code>	임계값 계산	신뢰구간 $\pm t$ 값
<code>cdf()</code>	누적확률	양측 검정 p-value
<code>sf()</code>	초과확률	단측 검정 p-value
<code>interval()</code>	신뢰구간 자동 계산	평균 $\pm t \times$ 표준오차

이 4개만 익숙해지면, t-검정, 신뢰구간, p-value 계산 대부분 처리 가능합니다.

이원분산분석 (Two-way ANOVA)

문제 : 크리스마스 장식 종류와 지역에 따라 판매량에 유의미한 차이가 있는지 이원 분산 분석을 통해 검정하세요!

크리스마스 장식 종류(트리, 조명, 장식품)가 판매량에 미치는 영향을 분석하세요. 이때, 장식 종류의 F-value, p-value를 구하시오
지역(북부, 남부, 동부, 서부)이 판매량에 미치는 영향을 분석하세요. 이때, 장식 종류의 F-value, p-value를 구하시오
크리스마스 장식 종류와 지역의 상호작용이 판매량에 미치는 영향을 분석하세요. 이때, 장식 종류의 F-value, p-value를 구하시오

- gpt 대화 링크
-

```
df=pd.read_csv('data/type3/christmas_decoration_sales.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Decoration_Type  36 non-null    object
1   Region          36 non-null    object
2   Sales           36 non-null    int64
dtypes: int64(1), object(2)
memory usage: 996.0+ bytes
```

```
# ols 다중선형회귀 분석 후 anova table 로 이원 분산 분석을 확인
from statsmodels.formula.api import ols
import statsmodels.api as sm
```

```
formula = 'Sales ~ C(Decoration_Type) + C(Region) + C(Decoration_Type):C(Region)'
model = ols(formula,df).fit()
sm.stats.anova_lm(model,typ=2)
```

	sum_sq	df	F	PR(>F)
C(Decoration_Type)	1764.500000	2.0	2.370578	0.114943
C(Region)	804.305556	3.0	0.720381	0.549614
C(Decoration_Type):C(Region)	5153.944444	6.0	2.308081	0.066915
Residual	8932.000000	24.0	NaN	NaN

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype
#   ...
```

```
-----
0  Decoration_Type  36 non-null    object
1  Region          36 non-null    object
2  Sales           36 non-null    int64
dtypes: int64(1), object(2)
memory usage: 996.0+ bytes
```

종속변수는 판매량, 독립변수는 장식 종류 + 지역 + 상호작용

- 일원분산분석
 - 독립변수(요인): 교육 방법 (A, B, C)
 - 종속변수: 시험 성적
- 이원분산분석
 - 독립변수(요인): 장식타입(트리,조명,장식품), 지역(동부,서부..) ... 2개 이상의 독립변수
 - 종속변수 : 판매량
- ANOVA도 결국 “집단에 따른 평균 차이” → 선형 회귀식으로 표현 가능
- ols()는 범주형 변수도 더미코딩해서 선형회귀 모델로 적합 가능

구분	목적	해석
OLS Regression Results	회귀계수 기반 → 예측력 중심 해석	각 독립변수가 타겟에 얼마나 영향을 미치는지 계수(+p-value)로 설명
anova_lm(model, typ=2)	분산 분석 기반 → 설명력 중심 해석	각 독립변수가 전체 분산 중 얼마나 기여했는지 분석 (F값 + p값)

typ	설명	사용 예
1	순서대로 변수 추가	모델 순서에 민감, 회귀분석 스타일
2	각 변수의 고유 기여도 (다른 변수 보정)	독립변수 간 상호작용 없을 때 적합
3	모든 변수 포함된 상태에서의 기여도 평가	더미 변수 포함 + 교호작용 있는 경우 권장

```
model.summary()
```

OLS Regression Results			
Dep. Variable:	Sales	R-squared:	0.464
Model:	OLS	Adj. R-squared:	0.218
Method:	Least Squares	F-statistic:	1.886
Date:	Wed, 18 Jun 2025	Prob (F-statistic):	0.0938
Time:	22:57:15	Log-Likelihood:	-150.33
No. Observations:	36	AIC:	324.7
Df Residuals:	24	BIC:	343.7
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
Intercept	61.6667	11.138	5.537	0.000	38.679	84.654
C(Decoration_Type)[T.조명]	24.0000	15.752	1.524	0.141	-8.510	56.510
C(Decoration_Type)[T.트리]	-19.0000	15.752	-1.206	0.239	-51.510	13.510
C(Region)[T.동부]	-14.0000	15.752	-0.889	0.383	-46.510	18.510
C(Region)[T.북부]	-17.6667	15.752	-1.122	0.273	-50.176	14.843
C(Region)[T.서부]	-3.3333	15.752	-0.212	0.834	-35.843	29.176

C(Decoration_Type)[T.조명]:C(Region)[T.동부]	-20.6667	22.276	-0.928	0.363	-66.642	25.309
C(Decoration_Type)[T.트리]:C(Region)[T.동부]	39.0000	22.276	1.751	0.093	-6.975	84.975
C(Decoration_Type)[T.조명]:C(Region)[T.북부]	13.6667	22.276	0.614	0.545	-32.309	59.642
C(Decoration_Type)[T.트리]:C(Region)[T.북부]	55.3333	22.276	2.484	0.020	9.358	101.309
C(Decoration_Type)[T.조명]:C(Region)[T.서부]	-26.0000	22.276	-1.167	0.255	-71.975	19.975
C(Decoration_Type)[T.트리]:C(Region)[T.서부]	36.6667	22.276	1.646	0.113	-9.309	82.642

Omnibus:	8.070	Durbin-Watson:	1.525
Prob(Omnibus):	0.018	Jarque-Bera (JB):	2.300
Skew:	0.058	Prob(JB):	0.317
Kurtosis:	1.767	Cond. No.	17.9

```
# dir(statsmodels.formula.api)
```

```
# dir(statsmodels.api.stats.anova_lm)
```

anova_lm()은 statsmodels의 핵심 분산분석 함수

```
df=pd.read_csv('data/type3/Customer_Data.csv')
```

```
df.head(2)
```

	age	income	marital_status	children	gender	purchase
0	62	111980	1	2	1	0
1	65	107314	0	3	1	0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   age             500 non-null   int64
1   income          500 non-null   int64
2   marital_status  500 non-null   int64
3   children        500 non-null   int64
4   gender          500 non-null   int64
5   purchase        500 non-null   int64
dtypes: int64(6)
memory usage: 23.6 KB
```

```
train= df.iloc[:350]
test=df.iloc[350:]
```



```
train.shape, test.shape
```

```
((350, 6), (150, 6))
```

```
import statsmodels.api as sm
import numpy as np
```

- import statsmodels.api as sm # ANOVA 표, 통계 모델 도우미
- import statsmodels.formula.api as smf # ols, logit 등 formula 모델 정의

이원분산분석 vs 다중선형회귀 차이

항목	다중선형회귀	이원분산분석 (Two-way ANOVA)
목적	수치형 종속변수를 여러 독립변수로 예측	집단 간 평균 차이 검정
독립변수 종류	수치형/범주형 모두 가능	범주형 2개 (예: 성별, 교육수준)
주요 함수	<code>ols(...).fit() + .summary()</code>	<code>ols(...)</code> + <code>sm.stats.anova_lm(...)</code>
출력 중심	회귀계수, p-value, R ² 등	각 요인의 F-값, p-value (주효과/상호작용 확인)
모델 구성 예시	<code>y ~ x1 + x2</code>	<code>y ~ A + B + A:B</code> (A:B 는 상호작용항)

잔차 이탈도

문제 : 잔차이탈도를 구하시오

- 고객 정보를 나타낸 데이터이다. 주어진 데이터에서 500개 중 앞에서부터 300개는 train으로, 200개는 test 데이터로 나눈다. 모델을 학습 (적합)할 때는 train 데이터를 사용하고, 예측할 때는 test 데이터를 사용한다. 모델은 로지스틱 회귀를 써서 고객이 특정 제품을 구매할지 여부를 예측하되, 페널티는 부과하지 않는다.

종속변수: purchase (0: 구매 안 함, 1: 구매 함)

Q. age, income, marital_status 변수를 독립변수로 purchase를 종속변수로 사용하여 로지스틱 회귀 모형을 만들고, 잔차이탈도를 구하시오. (반올림하여 소수 넷째자리까지 계산)

```
import pandas as pd
from statsmodels.formula.api import logit
```

```
df= pd.read_csv('data/type3/Customer_Data.csv')
```

```
train = df.iloc[:300]
test = df.iloc[300:]
```

```
model = logit('purchase ~ age + income + marital_status', data=train).fit()
```

```
Optimization terminated successfully.
Current function value: 0.687415
Iterations 4
```

```
print(model.summary())
```

Logit Regression Results						
=====						
Dep. Variable:	purchase	No. Observations:	300			
Model:	Logit	Df Residuals:	296			
Method:	MLE	Df Model:	3			
Date:	Wed, 18 Jun 2025	Pseudo R-squ.:	0.007474			
Time:	17:03:34	Log-Likelihood:	-206.22			
converged:	True	LL-Null:	-207.78			
Covariance Type:	nonrobust	LLR p-value:	0.3756			
=====						
	coef	std err	z	P> z	[0.025	0.975]

Intercept	0.3407	0.452	0.753	0.451	-0.546	1.227
age	-0.0094	0.008	-1.236	0.216	-0.024	0.005
income	3.636e-06	4.02e-06	0.904	0.366	-4.25e-06	1.15e-05
marital_status	-0.2192	0.235	-0.931	0.352	-0.680	0.242
=====						

```
model.params
```

```
Intercept      0.340700
age             -0.009367
income          0.000004
marital_status -0.219173
dtype: float64
```

Residual Deviance=-2·logL(β^)

항목	방향성	의미
로그 우도 11f	클수록 좋음	설명 잘함
잔차이탈도 -2 * 11f	작을수록 좋음	잔차 적음

```
# 잔차이탈도 -2 * 로그 우도 (Log-Likelihood) 모델의 적합도 지표
print(round(-2 * model.11f,2))
```

412.45

```
# summary 만 보고 계산하면 소수점이 모두 나오지 않기 때문에 차이가 있음
-2 * -206.22
```

412.44

다중 선형회귀

문제 :

- 모든 변수를 사용하여 OLS 모델을 적합하고, 회귀계수 중 가장 큰 값은?
- 유의미하지 않은 변수를 제거한 후 모델을 다시 적합하고, 회귀계수 중 가장 작은 변수명은?
- 2번 모델의 R-squared 값을 계산하고 해석하세요.
- 1번 모델에서 새로운 데이터(x1=5, x2=12, x3=10, x4=3)에 대해 y 값을 예측하세요.
- 1번 모델에서 x1, x2, x3, x4의 상관관계를 계산하고 가장 큰 상관계수를 구하시오. (단, 자기 상관관계 제외)
- x1과 x2만을 예측 변수로 사용하는 모델을 적합하고, 전체 모델과 R-squared 값을 구하시오.
- 잔차(residual) 분석을 수행하고, 잔차의 표준편차를 구하시오.
- 1번 모델에서 새로운 데이터(x1=5, x2=12, x3=10, x4=3)에 대해 y의 신뢰구간 하한(97% 신뢰수준)을 구하세요.
- 1번 모델에서 새로운 데이터(x1=5, x2=12, x3=10, x4=3)에 대해 y의 예측구간 상한(97% 신뢰수준)을 구하세요.

```
data = pd.read_csv('data/type3/t3_regression_data.csv')
```

```
data.head()
```

	x1	x2	x3	x4	y
0	1.354882	13.660146	12.025720	2.265464	-10.090372
1	8.878517	12.798070	14.194644	4.694667	7.798310
2	9.326056	18.815027	9.817460	2.519119	-2.287519
3	4.455682	11.667165	7.402825	4.801164	-1.945564
4	3.882355	11.008464	14.603228	3.360569	1.628605

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  -
 0    x1      100 non-null    float64
 1    x2      100 non-null    float64
 2    x3      100 non-null    float64
 3    x4      100 non-null    float64
 4     y      100 non-null    float64
dtypes: float64(5)
memory usage: 4.0 KB
```

```
# 필요한 라이브러리 импорт
import statsmodels.formula.api as smf

# 1. 모든 변수를 사용하여 OLS 모델 적합
model_full = smf.ols('y ~ x1 + x2 + x3 + x4', data=data).fit()
print("1. 전체 모델:")
print(model_full.summary())
```

1. 전체 모델:

OLS Regression Results			
=====			
Dep. Variable:	y	R-squared:	0.988
Model:	OLS	Adj. R-squared:	0.988
Method:	Least Squares	F-statistic:	1984.
Date:	Wed, 18 Jun 2025	Prob (F-statistic):	1.39e-90

```
Time: 17:19:49 Log-Likelihood: -130.88
No. Observations: 100 AIC: 271.8
Df Residuals: 95 BIC: 284.8
Df Model: 4
Covariance Type: nonrobust

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    2.9816      0.740      4.029      0.000      1.512      4.451
x1           1.9979      0.030     66.384      0.000      1.938      2.058
x2          -1.4924      0.034    -44.274      0.000     -1.559     -1.425
x3           0.4633      0.033     13.837      0.000      0.397      0.530
x4           0.0203      0.078      0.260      0.796     -0.135      0.175
=====

Omnibus: 2.161 Durbin-Watson: 1.921
Prob(Omnibus): 0.340 Jarque-Bera (JB): 1.718
Skew: 0.313 Prob(JB): 0.424
Kurtosis: 3.140 Cond. No. 155.
=====
```

Notes:

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

가장 무의미한 변수 x4 (p_value : 0.796)

```
# 2. 유의미하지 않은 변수를 제거한 후 모델 재적합
# 유의미한 변수만 골라 새로운 모델 적합
model_refit = smf.ols('y ~ x1 + x2 + x3', data=data).fit()
print("\n2. 유의미한 변수로 재적합한 모델:")
print(model_refit.summary())
```

2. 유의미한 변수로 재적합한 모델:

```
OLS Regression Results

=====
Dep. Variable: y R-squared: 0.988
Model: OLS Adj. R-squared: 0.988
Method: Least Squares F-statistic: 2672.
Date: Wed, 18 Jun 2025 Prob (F-statistic): 2.55e-92
Time: 17:27:00 Log-Likelihood: -130.91
No. Observations: 100 AIC: 269.8
Df Residuals: 96 BIC: 280.2
Df Model: 3
Covariance Type: nonrobust

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    3.0628      0.667      4.589      0.000      1.738      4.388
x1           1.9986      0.030     67.020      0.000      1.939      2.058
x2          -1.4942      0.033    -45.529      0.000     -1.559     -1.429
x3           0.4636      0.033     13.923      0.000      0.398      0.530
=====

Omnibus: 2.285 Durbin-Watson: 1.931
Prob(Omnibus): 0.319 Jarque-Bera (JB): 1.836
Skew: 0.324 Prob(JB): 0.399
Kurtosis: 3.140 Cond. No. 138.
```

Notes:

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

```
# 3. 초기 모델의 R-squared 값 계산
print("\n3. 초기 모델의 R-squared 값:", model_refit.rsquared)
```

3. 초기 모델의 R-squared 값: 0.9881643592045125

```
# 4. 새로운 데이터(x1=5, x2=12, x3=10, x4=3)에 대해 y 예측
new_data = pd.DataFrame({'x1': [5], 'x2': [12], 'x3': [10], 'x4': [3]})
y_pred = model_full.predict(new_data)
print("\n4. 새로운 데이터 포인트에 대한 예측값:", y_pred.iloc[0])
```

4. 새로운 데이터 포인트에 대한 예측값: -0.2433084582017389

```
# 5. 독립 변수 간의 상관관계 계산
correlation_matrix = data.corr()
print("\n5. 예측 변수 간의 상관관계 행렬:")
print(correlation_matrix)
```

5. 예측 변수 간의 상관관계 행렬:

	x1	x2	x3	x4	y
x1	1.000000	-0.115152	0.080456	0.117670	0.822594
x2	-0.115152	1.000000	-0.164049	-0.224881	-0.627445
x3	0.080456	-0.164049	1.000000	0.075652	0.301941
x4	0.117670	-0.224881	0.075652	1.000000	0.218884
y	0.822594	-0.627445	0.301941	0.218884	1.000000

```
# 6. x1과 x2만을 사용한 모델 적합 및 R-squared 값
model_x1_x2 = smf.ols('y ~ x1 + x2', data=data).fit()
print("\n6. x1과 x2만 사용한 모델의 R-squared 값:", model_x1_x2.rsquared)
```

6. x1과 x2만 사용한 모델의 R-squared 값: 0.9642665748957848

```
# 7. 잔차 분석 수행
residuals = model_full.resid
print("\n7. 잔차의 표준편차:")
print(residuals.std())
```

7. 잔차의 표준편차:
0.9001841451852484

```
# 8. 1번 모델에서 새로운 데이터에 대해 y의 신뢰구간 하한(97% 신뢰수준)을 구하세요.
pred_conf_int_97 = model_full.get_prediction(new_data).summary_frame(alpha=0.03)
conf_lower_97 = pred_conf_int_97['mean_ci_lower'].iloc[0]
print("\n8. y의 신뢰구간 하한(97% 신뢰수준):", conf_lower_97)
```

10. y의 신뢰구간 하한(97% 신뢰수준): -0.540230664436758

```
# pred = model.get_prediction(new_data).summary_frame(alpha=0.05)
# print(pred[['mean', 'mean_ci_lower', 'mean_ci_upper']])
```

예측값 + 신뢰구간/예측구간까지 함께 계산

시험에서 신뢰구간 하한/상한 물을 땀 반드시 사용해야 함

```
# 9. 1번 모델에서 새로운 데이터에 대해 y의 예측구간 상한(97% 신뢰수준)을 구하세요.
pred_upper_97 = pred_conf_int_97['obs_ci_upper'].iloc[0]
print("11. y의 예측구간 상한(97% 신뢰수준):", pred_upper_97)
```

11. y의 예측구간 상한(97% 신뢰수준): 1.802933556874063

```
pred_conf_int_97
```

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	-0.243308	0.13477	-0.540231	0.053614	-2.28955	1.802934

비모수 검정(wilcoxon)

문제 : 베스킨라빈스는 쿼트(Quart) 아이스크림의 중앙값이 620g이라고 주장하고 있습니다.

저는 실제로 이 아이스크림의 중앙값이 620g보다 무겁다고 주장합니다. 다음은 20개의 쿼트 아이스크림 샘플의 무게 측정 결과입니다. 이 측정 결과를 바탕으로 나의 주장이 사실인지 비모수 검정(Wilcoxon Signed-Rank Test)을 통해 검정해보십시오. p-value값을 반올림하여 소수점 둘째 자리까지 계산

귀무가설: "베스킨라빈스 쿼트 아이스크림의 중앙값은 620g이다."
대립가설: "베스킨라빈스 쿼트 아이스크림의 중앙값은 620g보다 무겁다."

```
import pandas as pd
data = {
    "weight": [630, 610, 625, 615, 622, 618, 623, 619, 620, 624, 616, 621, 617, 629, 626, 620, 618, 622, 625, 615,
               628, 617, 624, 619, 621, 623, 620, 622, 618, 625, 616, 629, 620, 624, 617, 621, 623, 619, 625, 618,
               622, 620, 624, 617, 621, 623, 619, 625, 618, 622]
}
df = pd.DataFrame(data)
```

항목	설명
모수 검정	정규분포, 평균, 분산 등을 가정하고 검정 (예: t-test)
비모수 검정	정규성 가정 없이 순위나 부호 등을 기반으로 검정
언제 사용?	샘플 수가 작거나, 정규분포를 따르지 않을 때

```
# 설정값
med = 620
```

```
df.median()
```



```
weight      621.0
dtype: float64
```

```
from scipy.stats import wilcoxon

wilcoxon(df['weight']- med,alternative='greater')
```

```
WilcoxonResult(statistic=684.0, pvalue=0.029661945180945556)
```

```
# help(wilcoxon)
```

함수	기준값 전달 방식
ttest_1samp()	popmean=620 ←  기준값 직접 인자 전달
wilcoxon()	기준값 없음  → 직접 차이값을 계산해서 넣어야 함

