<빅분기 실기_개념정리>

<1유형>데이터 전처리

- 데이터 타입 바꾸기 ⇒ *astype( )* 함수

```
df = df.astype({'cyl' : 'object})    #데이터 타입 1개 변경
```

```
df = df.astype({'cyl' :'int' ,'gear' :'object'})  #데이터 타입 2개 변경(cyl:int로 할당, gear:object로 할당)
```

- 데이터 **사분위수** 범위 구하기 ⇒ quantile(.25, .50, .75)

- 그룹별로 묶기 ⇒ **groupby() 함수**

```
변수명 = df.groupby('분류기준 1')['분류기준2'].함수( )
```

- 원하는 데이터 타입 선택하기 ⇒ *select_dtypes()* 함수

```
df.select_dtypes(include='데이터타입_종류')
```

- *str.contains()* 함수

- **##IQR** 구하기

```
q1 = df['feature'].quantile(0.25)
```

```
q3 = df['feature'].quantile(0.75)
```

```
iqr = q3 - q1
```

```
print(iqr)
```

- **이상치** 구하기(**Boxplot_상자그림** 기준)

```
upper = df[feature] + (1.5*iqr)
```

```
lower = df[feature] - (1.5*iqr)
```

```
cond1 = ( df[feature] ≤ upper )
```

```
cond2 = ( df[feature] ≥ lower )
```

```
outlier = (df[cond1] | df[cond2])
```

```
print(outlier)
```

- 데이터 **표준화, 정규화**하기 **#직접 구해서 계산하기 추천#**

  - **데이터 표준화(*Standard_Scaler*)**

```
mean = df['feature'].mean()
std = df['feature'].std()
z_score = (df['feature'] - mean) / std
```

  - **데이터 정규화*(MinMax_Scaler)***

```
max = df['feature'].max()
min = df['feature'].min()
minmax = (df['feature'] - min)  / (max - min)
```

- 첨도&왜도(kurtois, skewness)

  - *첨도(kurtois)*

```
kurt = df[feature].kurt()
print(kurt)
```

  - *왜도(skewness)*

```
skew = df[feature].skew()
print(skew)
```

- **데이터 indexing**

```
#  df.loc[행이름,열이름]
```

ex)
```
df.loc[3, 'mpg']    #행name: 3이고 열name: mpg인 df를 출력
df.loc[ : , mpg]    #행전체, 열name: mpg인 df를 출력
```

```
# df.iloc[행위치,열위치]
```

ex)
```
df.iloc[:,3:6]  #(전체 행, 3~5 번째 칼럼)을 필터링
```

- **데이터 filtering**

  ex_1) 'mtcars' 데이터 셋 에서 'cyl' 의 값이 4 인 데이터 수를 구하여라.

```
cond1 = (df['cyl'] = 4)
print(len(df[cond1]))
```

ex_2) 'mtcars' 데이터 셋 에서 'mpg'가 22 이상인 데이터 수를 구하여라.

```
cond2 = (df['mpg'] ≥ 22)
```

ex_3) 위의 조건을 **모두 만족**하는 데이터의 수를 구하여라.

```
print(len(df[cond1 & cond2]))  #2개조건 필터링
```

ex_4) 위의 조건 중 **하나라도 만족하는** 데이터의 수를 구하여라.

```
print(len(df[cond1 | cond2]))  #2개조건 필터링
```

- **데이터 정렬**

```
df.sort_values('mpg', ascending = False)  #내림차순 정렬
```

- **데이터 변경 (중요) ⇒ *np.where*( ) 활용**

```
np.where(조건,  조건만족할때의 값,  만족하지않을때 값)
```

ex) hp변수 값 중에서 205가 넘는 값은 205로, 나머지는 그대로 유지하여라.

```
df[hp] = np.where( df[hp]≥205, 205, df[hp])
```

- **결측치 대체(*fillna*( )함수 사용)**

```
median = df[feature].median()  #중앙값 구하기
df[feature] = df[feature].fillna(median)  #feature변수의 결측치를 중앙값으로 대체
df.isnull().sum() #잘 제거 되었는지 확인
```

- *결측치 제거(**dropna**(axis=0 or 1))*

```
df = df.dropna(axis=0) or (axis=1)   #각각 행기준, 열기준
```

- *중복값 제거* ⇒ drop.duplicate**s**()

```
df = df.drop_duplicates( )
print(df.shape)
```

- **데이터 합치기 (pd.concat( ) 함수) ⇒ ###중요!!###**

```
df_new = pd.concat (( df1 , df2 ), axis=0 ) # 행 방향으로 결합(위,아래)
print(df_new.head())
print(df_new.shape())
```

```
df_new = pd.concat((df1, df2), axis=1)  #열 방향으로 결합(좌, 우)
print(df_new.head())
```

- **시계열 데이터(Timeseries) ⇒ pd.to_datetime() 활용**

```
df ['날짜'] = pd.to_datetime (df['날짜'])  #데이터 타입을 datetime으로 변경
df['year'] = df['날짜'] .dt.year     #년,월,일 변수 추가하기
df['month'] = df['날짜']. dt.month
df['day'] = df['날짜']. dt.day
```

- **시계열 데이터 필터링(filtering)**

  1.between( ) 함수 사용

```
df[df[날짜].between(0000-00-00, 0000-00-00)] #좌우 모두 범위에 포함
```

  2. 날짜를 인덱스로 설정 후 loc( )함수 사용

```
df = df .set_index ('날짜',  drop=True(defalut)  or  False )

print(df.loc[0000-00-00 : 0000-00-00])  #두 가지 모두 결과는 같음
print( df.loc[(df.index ≥ 0000-00-00)  & (df.index  ≤ 0000-00-00)] )
```

  *## **특정시간대**를 **필터링** 해야할 때*

```
df.between_time(start_time=00:00:00, end_time=00:00:00)
```

- **index 다루기**

  ## index 새로 지정(컬럼을 인덱스로) ⇒ set_index()

```
df = df.set_index('time')
```

  ## index 초기화(인덱스를 컬럼으로) ⇒ reset_index()

```
df = df.reset_index()
```

- **map함수**

  *##타겟 변수(y)가 **범주형**(low,medium,high)이고 이것을 0,1,2로 바꿔야할때*

```
y[target] = y[target].map({low:0,  medium:1,  high:2})
```

- **replace함수**

```
df[feature] = df[feature].replace(바꿀대상,  바꿀값)  # 두 개 이상이면 리스트로 감싸기[ ]
ex) y[target] = y[target].replace(['low','medium','high'],[0,1,2])
```

#빅분기 시험환경 에서는 ***df.head()*** *함수도* ***print()적용*** *해야 보임*

```
ex) print(df.head())
```

- 반올림, **내림** 함수(round( ), int( ))

==========

<2유형>데이터 모델링

< base line code>

1. **데이터 탐색(EDA)**

```
df.head(), df.tail()  #데이터 앞부분 확인
df.shape  #(행, 열) 데이터 형태 확인
df.info()  #결측치 등 데이터 타입 확인
df.describe()  #통계값 분포
df.value_counts() #칼럼별 데이터 개수 확인
df.describe(include='object')
df.describe(include='category'))
```

2. *~~(이상치)~~,* ***결측치****,* ***feature****변수처리*

- 데이터 **결측치 확인**(***feature*** *기준*)

```
df.isnull().sum()  #결측치 확인
```

결측치처리 : ***범주형****(****최빈값-mode****),* ***연속형****(****중앙값-median****) ##*

- ***feature 변수처리***

```
df.info() #피처변수 데이터타입 확인
```

원핫인코더 (feature_male : 0 or 1, feature_female : 0 or 1) ⇒ 칼럼 수가 **늘어남** **(target에 활용금지)**

```
from sklearn.preprocessing import OneHotEncoder

oe = OneHotEncoder()  #pd.get_dummies도 사용가능

train = oe.fit_transform(train['feature']) #학습 데이터는 fit_transform()

test =  oe.fit_transform(test['feature'])  # 테스트 데이터는 transform()

# 원핫인코더는 열의 개수가 추가되므로 데이터 변수 전체에 할당
```

라벨인코더 (feature = 1,2,3...) ⇒ 숫자 형태로 분류 (칼럼 수가 늘어나지 않음)

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

train['feature'] = le.fit_transfrom(train['feature'])

test['feature'] = le.fit_transform(test['feature'])
```

#기타 : 불필요한 변수 제거

```
df = df.drop(columns='feature')  #(id, 중복되는 변수 등)
```

#제거할 변수가 여러 개일때 :

```
df = df.drop(columns=['feature1','feature2','feature3']) #리스트로 감싸기
```

4. 데이터 분할(train_test_split) ⇒ 분류 문제에서 **target변수**(y) **층화추출** 꼭 해주기 **(stratify = y)**

```
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.2, random_state=2024)

x_train, x_val, y_train, y_val = train_test_split(x, y['target'], test_size=0.2, random_state=2024)
```

5. 모델링(**랜덤포레스트** 분류 or회귀) (**훈련**, **검증 데이터셋** 활용)

```
from sklearn.ensemble import RandomForestClassifier / RandomForestRegressor

rfc/rfr = RandomForestClassifier / Regressor(random_state=2024)

ex) rfc.fit(x_train, y_train['target'])
```

6. (선택) **# *Hyperparameter 최적화(그리드 서치)*** *(훈련, 검증 데이터 셋 활용)* #

```
from sklearn.model_selection import GridSearchCV

param_grid = {'n_estimators':[30, 70, 100],

'max_depth:[6, 8, 10],

'min_samples_leaf' : [1, 2, 3] }
```

```
gridsearch = GridSearchCV( model, param_grid=param_grid, cv=5, n_jobs=-1 )
gridsearch.fit(x_train, y_train['target'])

print('best_hyperparameter:', gridsearch.best_params_)
print('best_score:', gridsearch_best_score_)
```

⇒*위의 최적화 파라미터로 **##Test_dataset##** 랜덤포레스트 수행 (**# GridSearchCV 부분 주석 취할 것**)*

7. 성능평가 (**주의**: 분류 문제에서 분류될 확률을 구하는 문제는 **predict_proba(y_test, pred))**

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import mse, r2_score
```

8. 답안 제출 (filiename.to_csv('xxxxx.csv', **index=False**)로 제출, read_csv()로 확인)

==========

<3유형>가설검정과 회귀분석

3-1. 가설검정

<#1>모집단 1개 - 단일표본 t검정(1 sample t-test) (집단의 평균, 특정 값)

Q1. mtcars 데이터셋의 mpg열의 데이터 **평균**이 20과 **같다고 할 수 있는지** 검정하시오. (유의수준 5%)

<가설설정>

H0 : mpg열의 평균이 20과 **같다**.

H1 : mpg열의 평균이 20과 **같지 않다**.

t-test전에 정규성 검정(shapiro)검정 ⇒ 정규성 만족시 1samp_ttest ,

⇒ 정규성 만족x시 wilcoxon 부호순위검정 실시

##정규성 검정##

H0 : 정규분포를 따른다.

H1 : 정규분포를 따르지 않는다.

```
import scipy.stats as stats
from scipy.stats as shapiro
```

```
statistic, pvalue = stats.shapiro(df['mpg'])
print(round(statistic, 4), round(pvalue, 4))
```

`#statistic=0.9476, p_value=0.1229` 이므로

 # 귀무가설 채택(정규분포를 따른다.)

(**정규성** 만족) 단일표본 t 검정 실시 ⇒ (**ttest_1samp**)

```
statistic, pvalue = stats,ttest_1samp(df['mpg'], popmean=20, alternative ='two-sided'  )
print(round(statistc,4) , round(pvalue, 4))
```

`#statistic=0.0851, pvalue=0.7891` 이므로

#귀무가설 채택 (mpg열의 평균이 20과 **같다**고 할 수 있다.)

(**정규성** 만족x) 윌콕슨 부호순위검정 실시 ⇒ (**wilcoxon**)

```
statistic, pvalue = stats.wilcoxon(df['mpg'] - 17, alternative = 'two-sided')
print(round(statistic, 4), round(pvalue,4))
```

`#statistic=249.0, pvalue=0.7891` 이므로

#귀무가설 채택 (mpg열의 평균이 20과 같다고 할 수 있다.)

<#2>모집단 2개

***각 검정함수의 파라미터 형태,위치 기억해둘것 ***

1. (정규성 O) 대응표본(쌍체) t-검정 ⇒ paired t-test (stats.ttest_rel)

(정규성 X)윌콕슨 부호순위 검정 ⇒ wilcoxon

2. (정규성 O) 독립표본 t검정 ⇒ 2sample t-test (stats.ttest_ind)

(정규성 X)윌콕슨 순위합 검정(ranksums)

1. 대응 표본(쌍체) t검정

- **정규성 검정** (**차이값**에 대한 정규성 검정) ⇒ `stats.shapiro(df['after'] - df['before'])`

  `H0 : after - before = 0`

  `H1 : after - before ≠ 0`

- **대응(쌍체)표본 검정 실시:** `stats.ttest_rel(df['after'], df['before'], alternative = 'two-sided')`

  *ex) 혈압약을 먹은 후 **혈압이 감소했는지** 확인하기 위해 **쌍체 t검정**을 실시하시오.*

  H0 : after - before ≥ 0 *(혈압이 **감소하지 않았다.**)*

  H1 : after - before < 0 *(혈압이 **감소하였다.**)*

2. 독립표본 t검정

- **정규성 검정**(**두 집단 모두 정규성을 따르는지**) ⇒ `stats.shapiro(['A']) , stats.shapiro(['B'])`
- **등분산 검정** ⇒ `stats.bartlett(df['A'], df['B'])`

  등분산 여부 확인 필수 (**equal_var**=**True** or **False**)

  (등분산 만족O) `stats.ttest_ind(df['A'], df['B'], equal_var=True , alternative='two-sided')`

  (등분산 만족X) `stats.ttest_ind(df['A'], df['B'], equal_var=False , alternative='two-sided')`

- **독립표본 t 검정 실시 :** `stats.ttest_ind(df['A'], df['B'], equal_var=True, alternative='two-sided')`

  *ex) 두 그룹의 혈압평균이 **다르다**고 할 수 있는지 **독립표본 t 검정**을 실시하시오.*

  H0 : A = B (혈압평균이 같다)

  H1 : A ≠ B (혈압평균이 다르다)

<#3>모집단 3개 이상- 분산분석(ANOVA) : A집단 vs B집단 vs C집단.....

(정규성 만족o) - **ANOVA 분석 (stats.f_oneway) # 데이터가 각각 들어가야 함**

(정규성 만족x) - 크루스칼-왈리스 검정(kruskal - wallis test)

ex) *다음 A,B,C 그룹의 성적 평균이 같다고 할 수 있는지 ANOVA분석을 실시하시오.*

H0 : A=B=C(세 그룹의 평균이 **모두 같다**.)

H1 : Not H0 (**적어도 하나는** 같지 않다.)

- **정규성 검정 ⇒** `statistic, pvalue = (stats.shapiro( df['A'] ))` ****각 정규성 검정 진행**

  `statistic, pvalue = (stats.shapiro( df['B'] ))`

  `statistic, pvalue = (stats.shapiro( df['C'] ))`
- **등분산성 검정 ⇒** `statistic, pvalue = stats.bartlett(df['A'], df['B'], df['C'])`

  ****등분산 여부 확인(만족x시 크루스칼-왈리스 검정)**
- **ANOVA분석 ⇒** `statistic, pvalue = stats.f_oneway(df['A'], df['B'], df['C'])`

<#3> 카이제곱 검정

1. 적합도 검정 - 각 범주에 속할 확률이 같은지 - **chisquare()**
2. 독립성 검정 - 두 개의 범주형 변수가 서로 독립인지 - **chi2_contingency()**

1-1. 적합도 검정 예시 문제 - example_1

ex) 랜덤 박스에 상품이 들어있다. 다음은 랜덤박스 에서 100번 상품을 꺼냈을 때의 상품 데이터라고 할 때,

상품이 동일한 비율로 들어있다고 할 수 있는지 검정해보시오. (유의수준 5%)

```
import pandas as pd
import numpy as np

#데이터 생성
rowl = [30, 20, 15, 35]
df = pd.DataFrame([rowl], columns=['A','B','C','D'])
```

#가설설정

H0 : 랜덤박스에 상품 A,B,C,D가 동일한 비율로 들어있다.

H1 : 랜덤박스에 상품 A,B,C,D가 동일한 비율로 들어있지 않다.

```
from scipy.stats as import chisquare

f_obs = [30, 20, 15, 35]
f_exp = [25, 25, 25, 25]    #관측빈도와 기대빈도 구하기

statistic, pvalue = chisquare(f_obs=f_obs, f_exp=f_exp)
print(statistic, pvalue)
#statistic = 10.0, p_value= 0.0185661354….로 귀무가설 기각(대립가설 채택)
```

1-2. 적합도 검정 예시 문제 - example_2

ex) 랜덤 박스에 상품이 들어있다. 다음은 랜덤박스 에서 150번 상품을 꺼냈을 때의 상품 데이터라고 할 때,

상품별로 A: 30%, B:15%, C:55% 비율로 들어있다고 할 수 있는지 검정해보시오. (유의수준 5%)

```
import pandas as pd
import numpy as np

row1 = [50,25,75]
df = pd.DataFrame([row1], columns=['A','B','C'])
```

#가설설정

H0 : 랜덤박스에 상품 A, B, C가 30%, 15%, 55% 의 비율로 들어있다.

H1 : 랜덤박스에 상품 A, B, C가 30%, 15%, 55% 의 비율로 들어있지 않다.

```
#검정실시
from scipy.stats import chisquare

f_obs = [50, 25, 75]
a = 150*0.3
b = 150*0.15
c = 150*0.55

f_exp = [a, b, c]

statistic, pvalue = chisquare(f_obs=f_obs, f_exp=f_exp)
print(statistic, pvalue)
# statistic = 1.5151515151…. , p_value = 0.4688015…..
#귀무가설 채택(대립가설 기각)
```

2-1. 독립성 검정 예시 문제 - example_1

연령대에 따라 먹는 아이스크림의 차이가 있는지 독립성 검정을 실시하시오.

```
import pandas as pd
import numpy as np

row1, row2 = [200, 190, 250],  [220, 250, 300]
df = pd.DaraFrame([row1, row2], columns=['딸기', '초코', '바닐라'], index = ['10대', '20대'])
```

#가설설정

H0 : 연령대와 먹는 아이스크림의 종류는 서로 관련이 없다(두 변수는 서로 독립이다)

H1 : 연령대와 먹는 아이스크림의 종류는 서로 관련이 있다(두 변수는 서로 독립이 아니다)

```
#검정실시
from scipy.stats import chi2_contingency
statistic, pvalue, dof, expected = chi2_contingency(df)

print(statistic, pvalue, dof, np.round(expected, 2))
#1.708360….
0.4256320……
2
[[190.64  199.72  249.65]
[229.36  240.28  300.35]]
```

cf) 만약 데이터 형태가 다른 경우 - `pd.crosstab()` 사용

```
df = pd.DataFrame({
‘아이스크림’ : ['딸기', '초코', '바닐라', '딸기', '초코', '바닐라'],
‘연령’ : ['10대', '10대', '10대', '20대', '20대', '20대'],
‘인원’ : [200,190,250,220,250,300]
})

# pd.crosstab(index=   , columns=    , values=      , aggfunc=sum)
table = pd.crosstab(index=df[’연령’], columns=df[’아이스크림], values=df[’인원’], aggfunc=sum)

from scipy.stats import chi2_contingency

statistic, pvalue, dof, expected = chi2_contingency(table)
print(statistic, pvalue, dof, expected)

#1.708360….
0.4256320……
2
[[190.64  199.72  249.65]
[229.36  240.28  300.35]]
```

2-2. 독립성 검정 예시 문제 - example_2

타이타닉 데이터에서 성별(sex)과 생존여부(survived) 변수간 독립성 검정을 실시하시오.

```
import seaborn as sns
df = sns.load_dataset('titanic')
```

```
table = pd.crosstab(df['sex], df['survived])
print(table)
```

#가설설정

H0 : 성별과 생존여부는 서로 관련이 없다(두 변수는 서로 독립이다)

H1 : 성별과 생존여부는 서로 관련이 있다(두 변수는 서로 독립이 아니다)

```
# 검정실시(통계량, p-value, 기대빈도 확인)
from scipy.stats import chi2_contingency
statistic, pvalue, dof, expected = chi2_contingency(table)
print(statistic, pvalue, dof, np.round(expected))
```

```
#260.71702016732104
#1.1973570627755645e-58
#1
#[[193.47  120.53]
  [355.53  221.47]]
```

3-2. 회귀 분석

다중회귀분석

<module>

(1) `sklearn.linear_model의 LinearRegression 모델`

(2) `statsmodel.api(sm)의 OLS 모델`

(1) linear model code : scikit-learn 선형회귀 모델 코드 예시

```
x = [['feature1','feature2', 'feature3']]
y = df['target']

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x, y)

#결정계수 : model.score(x,y)
#회귀계수 : model.coef_
#회귀절편 : model.intercept_
```

(2) statsmodels code : sm.OLS 모델 코드 예시

```
x = [['feature1','feature2', 'feature3']]
y = df['target']

import statsmodels.api as sm

x = sm.add_constant(x)
model = sm.OLS(y, x).fit()
summary = model.summary()
print(summary)
```

상관분석

```
x = df['feature']
y = df['target']
```

```
from scipy.stats import pearsonr
corr, pvalue = pearsonr(x, y)
```

```
H0 : 두 변수간 선형관계가 존재하지 않는다.
H1 : 두 변수간 선형관계가 존재한다.
```

```
print(corr) #상관계수 출력
print(pvalue) # pvalue 출력
```

```
n = len(x) # 데이터 수
r2 = r**2  #상관계수의 제곱
statistic = r * ((n-2)**0.5) / ((1-r2)**0.5)

print(round(statistic, 2)) #통계량 출력
```

```
if statistic < 0.05 :
print('귀무가설 기각: 선형관계가 존재한다.')
else :
print('귀무가설 채택 : 선형관계가 존재하지 않는다.')
```

로지스틱 회귀분석

<module>

(1) `sklearn.linear_model의 LogisticRegression 모델`

(2) `statsmodel.api(sm)의 Logit 모델`

(1) linear model code : scikit-learn 로지스틱 회귀 모델 코드 예시

```
x = df.drop(columns='target')
y = df['target']
```

```
from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression (penalty=None)
```

```
model = lr.fit(x, y)
```

```
print(model.coef_)  #회귀계수 출력
print(model.intercept_)  #회귀절편 출력
```

(2) statsmodels code : sm.OLS 모델 코드 예시

```
import statsmodels.api as sm
x = sm.add_constant(x)
model = sm.Logit(y, x).fit()
summary = model.summary()
print(summary)
```

3과목 중요 기출 문제 keyword

```
로지스틱 회귀분석 (penalty = None)
statsmodels 는 상수항 꼭 추가 (x = sm.add_constant(x))
statsmodels 는 학습할때  (y, x) 위치 반대로 입력   (sm.Logit(y,x).fit())
회귀 계수 : model.coef_
회귀 절편 : model.intercept_
리스트 형식의 변수를 반올림할 때: np.round()
```

3과목 중요 기출 문제

Q1 . 로지스틱 회귀모형에서 ***sibsp 변수가 한단위 증가할 때 생존할 오즈가 몇 배 증가하는지*** *반올림하여*

소수점 셋째자리까지 구하시오

Q2. 로지스틱 회귀모형에서 ***여성일 경우 남성에 비해 오즈가 몇 배 증가하는지*** *반올림하여*

소수점 셋째 자리까지 구하시오.

Q3. 로지스틱 회귀분석 진행후 ***유의하지 않은 변수의 수****는?*

Q4. ***유의한 변수로만*** *로지스틱회귀분석을 진행한 후* ***회귀계수의 합계를 구하라(절편도 포함)****.*

Q5. 모델링 후 p-value가 가장 작은 변수의 ***회귀계수*** *값을 구하시오.*

Q6. 위에서 구한 다중선형회귀 모델의 ***결정계수(R-square)****를 구하시오.*

Q7. 위에서 만든 모델에 ***A=000, B=000, C=000값을 대입****하면* ***Y값은 얼마로 예측되겠는가****?*

Q8. 초기 500개 데이터로 분석시 sibsp 변수의 ***odds_ratio****는?*

Q9. 초기 500개 데이터로 분석시 ***residual_deviance(잔차이탈도)****는?*

Q10. 나머지 391개 데이터 적용시 ***오분류율은****?*

Q11. target 변수와 ***가장 큰 상관관계를 갖는*** *변수의* ***상관계수****를 구하시오.*

Q12. 다중선형회귀 모델링 후 ***결정계수(r2_score)****를 구하시오.*

Q13. 위에서 구한 회귀모델에서 ***p-value가 가장 큰 변수****의* ***p-value값****을 구하시오.*