In [5]:
```
import pandas as pd
import numpy as np
```

In [9]:
```
df = pd.read_csv("lv2-2305-1.csv")
```

In [10]:
```
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
```

In [36]:
```
df5=df.copy()
```

In [37]:
```
df5.head(2)
```

Out[37]:

	EMP_ID	NUM_LOGIN	NUM_CONTENTS	NUM_ACTION	TIME_LOGIN	TIME_TEST	NUM_REVISION	SCORE
0	S0001	7	10	57	150	1.95	1	9.77
1	S0002	7	11	54	144	2.42	10	5.80

In [38]:
```
q1 = df5['SCORE'].quantile(0.25)
q3 = df5['SCORE'].quantile(0.75)
```

In [39]:
```
q1, q3
```

Out[39]:
```
(5.38, 7.6899999999999995)
```

In [40]:
```
iqr = q3-q1
df5_1 = df5[(df5['SCORE'] >= q1 -1.5*iqr) & (df5['SCORE'] <= q3 + 1.5*iqr)]
```

In [41]:

```
df5_1['EMP_ID']
```

Out[41]:

```
0      S0001
1      S0002
2      S0003
3      S0004
4      S0005
       ...
263    S0264
264    S0265
265    S0266
266    S0267
267    S0268
Name: EMP_ID, Length: 267, dtype: object
```

In [42]:

```
train = df5_1[df5_1['EMP_ID'].str[4].astype('int') % 4 != 0]
test = df5_1[df5_1['EMP_ID'].str[4].astype('int') % 4 == 0]
```

In [43]:

```
y = train[['SCORE']]
max_num = -999
f_list= ['NUM_LOGIN', 'NUM_CONTENTS', 'NUM_ACTION', 'TIME_LOGIN', 'TIME_TEST', 'NUM_REVISION']
for f in f_list:
    x = train[[f]]
    lr = LinearRegression()
    lr.fit(x,y)
    adj_r = 1 - (1-lr.score(x,y))*(len(y)-1)/(len(y)-x.shape[1]-1)
    if adj_r > max_num:
        max_num = adj_r
        max_var = f
print(max_num, max_var)
```

```
0.26585408702965907 TIME_LOGIN
```

In [44]:

```
y = train[['SCORE']]
max_num = -999
f_list= ['NUM_LOGIN', 'NUM_CONTENTS', 'NUM_ACTION', 'TIME_TEST', 'NUM_REVISION']
for f in f_list:
    x = train[[f,'TIME_LOGIN']]
    lr = LinearRegression()
    lr.fit(x,y)
    adj_r = 1 - (1-lr.score(x,y))*(len(y)-1)/(len(y)-x.shape[1]-1)
    if adj_r > max_num:
        max_num = adj_r
        max_var = f
print(max_num, max_var)
```

```
0.29674428713721424 NUM_CONTENTS
```

In [46]:

```
y = train[['SCORE']]
max_num = -999
f_list= ['NUM_LOGIN', 'NUM_ACTION', 'TIME_TEST', 'NUM_REVISION']
for f in f_list:
    x = train[[f,'TIME_LOGIN','NUM_CONTENTS']]
    lr = LinearRegression()
    lr.fit(x,y)
    adj_r = 1 - (1-lr.score(x,y))*(len(y)-1)/(len(y)-x.shape[1]-1)
    if adj_r > max_num:
        max_num = adj_r
        max_var = f
print(max_num, max_var)
```

```
0.3265951102870196 NUM_REVISION
```

In [51]:

```
from sklearn.metrics import mean_squared_error
train_x = train[['TIME_LOGIN', 'NUM_CONTENTS', 'NUM_REVISION']]
train_y = train[['SCORE']]
test_x = test[['TIME_LOGIN', 'NUM_CONTENTS', 'NUM_REVISION']]
test_y = test[['SCORE']]
# Sklearn
lr = LinearRegression()
lr.fit(train_x,train_y)

test_y.loc[:,'prd'] = lr.predict(test_x)
res = mean_squared_error(test_y['prd'], test_y['SCORE']) **0.5
```

```
C:₩Users₩Hyemin₩AppData₩Local₩Temp₩ipykernel_26044₩492564208.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-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  test_y.loc[:,'prd'] = lr.predict(test_x)
```

In [52]:

```
res
```

Out[52]:

```
1.3919187141780964
```

In [ ]:

```
```

In [ ]:

```
#3번
#의사결정 나무란 : 나무의 가지(branch)처럼 데이터가 "예" 또는 "아니오" 같은 질문에 따라 나뉘고, 마지막에는 어떤 결론(예측)에 도달
#랜덤포레스트란 : 하나의 트리만 사용하면 그 트리의 성능이 데이터에 너무 의존하게 될 수 있습니다.
#                여러 개의 트리를 만드는데, 각 트리의 예측 결과를 다수결 투표나 평균을 내어 최종 예측을 만듭니다.
#                예를 들어, 100개의 트리가 있고 그중 60개가 "합격"을 예측하고, 40개가 "불합격"을 예측하면,
#                랜덤포레스트는 "합격"을 최종 예측으로 선택합니다.
```

In [54]:

```
df6 = df.copy()
```

In [55]:

```
# Step 3-1
df6_a = df6[df6['SCORE'] <= 5]
df6_b = df6[(df6['SCORE'] <=7) & (df6['SCORE']>5)]
df6_c = df6[(df6['SCORE'] <=10) & (df6['SCORE']>7)]
```

In [56]:

```
df6.head()
```

Out[56]:

	EMP_ID	NUM_LOGIN	NUM_CONTENTS	NUM_ACTION	TIME_LOGIN	TIME_TEST	NUM_REVISION	SCORE
0	S0001	7	10	57	150	1.95	1	9.77
1	S0002	7	11	54	144	2.42	10	5.80
2	S0003	2	11	21	132	2.14	1	9.77
3	S0004	10	11	101	350	2.73	7	7.15
4	S0005	4	11	35	93	1.61	5	5.28

In [57]:

```
from sklearn.ensemble import RandomForestRegressor

# Step 3-2
rf_a = RandomForestRegressor(random_state=1234, n_estimators = 10, min_samples_leaf = 10)
rf_b = RandomForestRegressor(random_state=1234, n_estimators = 10, min_samples_leaf = 10)
rf_c = RandomForestRegressor(random_state=1234, n_estimators = 10, min_samples_leaf= 10)

train_a_y = df6_a['SCORE']
train_a_x = df6_a.drop(columns=['EMP_ID','SCORE'])

train_b_y = df6_b['SCORE']
train_b_x = df6_b.drop(columns=['EMP_ID','SCORE'])

train_c_y = df6_c['SCORE']
train_c_x = df6_c.drop(columns=['EMP_ID','SCORE'])


#3-3

rf_a.fit(train_a_x, train_a_y)
a_fi = rf_a.feature_importances_

rf_b.fit(train_b_x, train_b_y)
b_fi = rf_b.feature_importances_

rf_c.fit(train_c_x, train_c_y)
c_fi = rf_c.feature_importances_
```

In [15]:

```
# Step 3-3 Compare Feature Importance
```

In [61]:
```
train_a_x.head()
```

Out[61]:

	NUM_LOGIN	NUM_CONTENTS	NUM_ACTION	TIME_LOGIN	TIME_TEST	NUM_REVISION
8	9	11	64	91	2.53	13
10	2	2	16	28	1.73	10
16	10	11	93	144	2.26	17
19	4	1	22	58	2.24	2
20	1	1	5	10	1.00	0

In [16]:
```
a_fi
```

Out[16]:
```
array([0.1       , 0.28329769, 0.6       , 0.        , 0.        ,
       0.01670231])
```

In [17]:
```
b_fi
```

Out[17]:
```
array([0.02380331, 0.00271996, 0.03890917, 0.74317753, 0.16952741,
       0.02186262])
```

In [18]:
```
c_fi
```

Out[18]:
```
array([0.        , 0.0074127 , 0.17934068, 0.10480842, 0.6058609 ,
       0.1025773 ])
```

In [63]:
```
import numpy as np

np.argmax(a_fi), np.argmax(b_fi), np.argmax(c_fi)
```

Out[63]:
```
(2, 3, 4)
```

In [19]:
```
res = a_fi.max() + b_fi.max() + c_fi.max()
```

In [20]:

```
print('Q3 Result:', res)
print('Q3 Answer:', round(res, 2))
```

```
Q3 Result: 1.9490384300673917
Q3 Answer: 1.95
```

In [ ]: