술은 공부에 얼마나 영향을 미칠까?

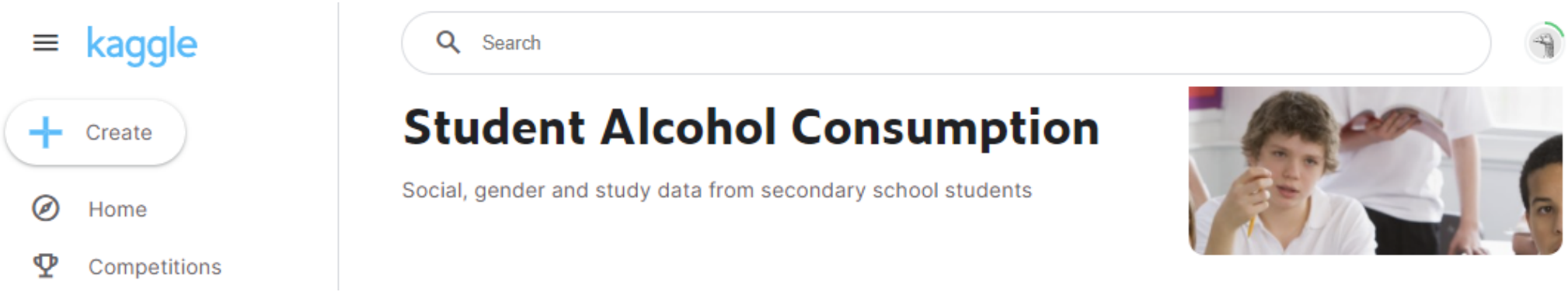
사물인터넷학과 20181537 명o성
사물인터넷학과 20201500 박o은
사물인터넷학과 20201514 이o수
사물인터넷학과 20201532 노o정

Index

Why?

술자리가 많은 대학생들 중 성적에 대한 걱정을 하는 대학생 다수

kaggle에서 'Student Alcohol Consumption' 이라는 흥미로운 데이터셋 발견



이를 통해 '성적 예측 프로그램'을 만들어 보고자 함.

https://www.kaggle.com/datasets/uciml/student-alcohol-consumption

Columns

1. school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2. sex - student's sex (binary: 'F' - female or 'M' - male)
3. age - student's age (numeric: from 15 to 22)
4. address - student's home address type (binary: 'U' - urban or 'R' - rural)
5. famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6. Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7. Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
8. Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9. Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10. Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
11. reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12. guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13. traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14. studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15. failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16. schoolsup - extra educational support (binary: yes or no)
17. famsup - family educational support (binary: yes or no)
18. paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19. activities - extra-curricular activities (binary: yes or no)
20. nursery - attended nursery school (binary: yes or no)
21. higher - wants to take higher education (binary: yes or no)
22. internet - Internet access at home (binary: yes or no)
23. romantic - with a romantic relationship (binary: yes or no)
24. famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25. freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26. goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27. Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28. Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29. health - current health status (numeric: from 1 - very bad to 5 - very good)
30. absences - number of school absences (numeric: from 0 to 93)

1. G1 - first period grade (numeric: from 0 to 20)
2. G2 - second period grade (numeric: from 0 to 20)
3. G3 - final grade (numeric: from 0 to 20, output target)

Model Selection Standard

지도 학습
- 레이블 된 훈련 샘플 존재

회귀 문제
- 값 예측 필요

다중 회귀 문제
- 예측에 사용할 특성 여러 개 존재

단변량 회귀 문제
- 하나의 값 도출 필요

배치 학습
- 실시간 학습 불필요

Data Preprocessing

```
# 필요없는 컬럼 삭제
studentACU = studentAC.drop(['school','Pstatus', 'Medu','Fedu','Mjob','Fjob','reason','guardian','paid','nursery'], axis=1)

studentACU
```

	sex	age	address	famsize	traveltime	studytime	failures	schoolsup	famsup	activities	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	F	18	U	GT3	2	2	0	yes	no	no	...	4	3	4	1	1	3	6	5	6	6
1	F	17	U	GT3	1	2	0	no	yes	no	...	5	3	3	1	1	3	4	5	5	6
2	F	15	U	LE3	1	2	3	yes	no	no	...	4	3	2	2	3	3	10	7	8	10
3	F	15	U	GT3	1	3	0	no	yes	yes	...	3	2	2	1	1	5	2	15	14	15
4	F	16	U	GT3	1	2	0	no	yes	no	...	4	3	2	1	2	5	4	6	10	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1039	F	19	R	GT3	1	3	1	no	no	yes	...	5	4	2	1	2	5	4	10	11	10
1040	F	18	U	LE3	1	2	0	no	yes	no	...	4	3	4	1	1	1	4	15	15	16
1041	F	18	U	GT3	2	2	0	no	no	yes	...	1	1	1	1	1	5	6	11	12	9
1042	M	17	U	LE3	2	1	0	no	no	no	...	2	4	5	3	4	2	6	10	10	10
1043	M	18	R	LE3	3	1	0	no	no	no	...	4	4	1	3	4	5	4	10	11	11

1044 rows × 23 columns

Data Preprocessing

object 타입을 int64형으로 바꾸기

```
studentACU['sex'] = studentACU['sex'].map({'F': 0, 'M':
studentACU['address'] = studentACU['address'].map({'U':
studentACU['famsize'] = studentACU['famsize'].map({'LE3
#(이진수: 'LE3' - 3 이하 또는 'GT3' - 3 초과)

studentACU = studentACU.replace('yes', 1) #famsup, acti
studentACU = studentACU.replace('no', 0)

studentACU
```

```
studentACU.info() # int 타입으로 잘 바뀌었는지 확인
```

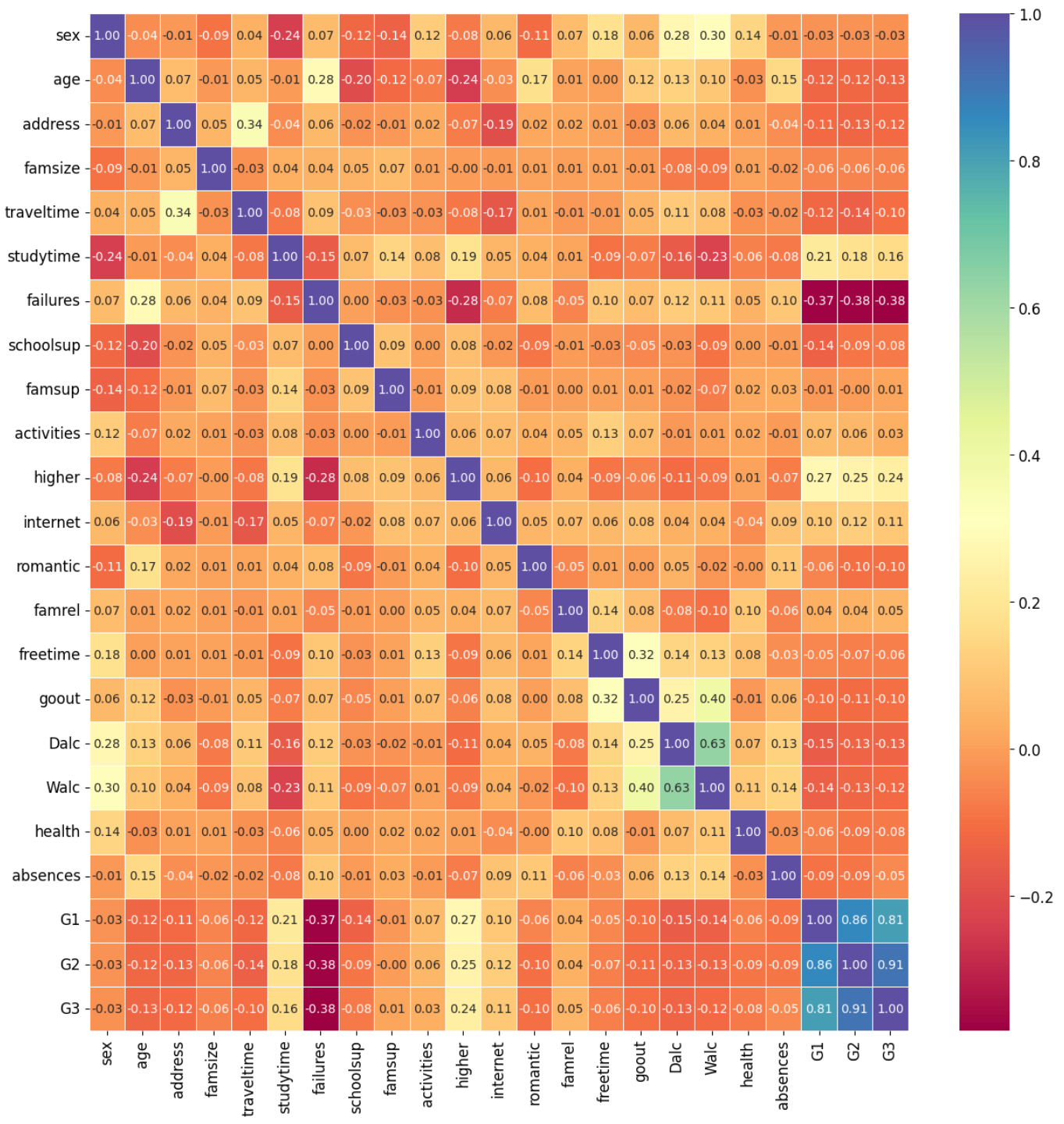
```
Output exceeds the size limit. Open the full output data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1044 entries, 0 to 1043
Data columns (total 23 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   sex         1044 non-null   int64
 1   age         1044 non-null   int64
 2   address     1044 non-null   int64
 3   famsize     1044 non-null   int64
 4   traveltime  1044 non-null   int64
 5   studytime   1044 non-null   int64
 6   failures    1044 non-null   int64
 7   schoolsup   1044 non-null   int64
 8   famsup      1044 non-null   int64
 9   activities  1044 non-null   int64
 10  higher      1044 non-null   int64
 11  internet    1044 non-null   int64
 12  romantic    1044 non-null   int64
 13  famrel      1044 non-null   int64
 14  freetime    1044 non-null   int64
 15  goout       1044 non-null   int64
 16  Dalc        1044 non-null   int64
 17  Walc        1044 non-null   int64
 18  health      1044 non-null   int64
 19  absences    1044 non-null   int64
...
 21  G2          1044 non-null   int64
 22  G3          1044 non-null   int64
dtypes: int64(23)
memory usage: 187.7 KB
```

---> 한번씩만 실행하셈
변환 -----> 한번씩만 실행하셈
--> 한번씩만 실행하셈

	sex	age	address	famsize	traveltime	studytime	fail	time	goout	Dalc	Walc	health	absences	G1	G2	G3
0	0	18	0	4	2	2		3	4	1	1	3	6	5	6	6
1	0	17	0	4	1	2		3	3	1	1	3	4	5	5	6
2	0	15	0	2	1	2		3	2	2	3	3	10	7	8	10
3	0	15	0	4	1	3		2	2	1	1	5	2	15	14	15
4	0	16	0	4	1	2		3	2	1	2	5	4	6	10	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1039	0	19	1	4	1	3		4	2	1	2	5	4	10	11	10
1040	0	18	0	2	1	2		3	4	1	1	1	4	15	15	16
1041	0	18	0	4	2	2		1	1	1	1	5	6	11	12	9
1042	1	17	0	2	2	1		4	5	3	4	2	6	10	10	10
1043	1	18	1	2	3	1		4	1	3	4	5	4	10	11	11

1044 rows × 23 columns

Heat Map


sex 1.00 -0.04 -0.01 -0.09 0.04 -0.24 0.07 -0.12 -0.14 0.12 -0.08 0.06 -0.11 0.07 0.18 0.06 0.28 0.30 0.14 -0.01 -0.03 -0.03 -0.03
age -0.04 1.00 0.07 -0.01 0.05 -0.01 0.28 -0.20 -0.12 -0.07 -0.24 -0.03 0.17 0.01 0.00 0.12 0.13 0.10 -0.03 0.15 -0.12 -0.12 -0.13
address -0.01 0.07 1.00 0.05 0.34 -0.04 0.06 -0.02 -0.01 0.02 -0.07 -0.19 0.02 0.02 0.01 -0.03 0.06 0.04 0.01 -0.04 -0.11 -0.13 -0.12
famsize -0.09 -0.01 0.05 1.00 -0.03 0.04 0.04 0.05 0.07 0.01 -0.00 -0.01 0.01 0.01 0.01 -0.01 -0.08 -0.09 0.01 -0.02 -0.06 -0.06 -0.06
traveltime 0.04 0.05 0.34 -0.03 1.00 -0.08 0.09 -0.03 -0.03 -0.03 -0.08 -0.17 0.01 -0.01 -0.01 0.05 0.11 0.08 -0.03 -0.02 -0.12 -0.14 -0.10
studytime -0.24 -0.01 -0.04 0.04 -0.08 1.00 -0.15 0.07 0.14 0.08 0.19 0.05 0.04 0.01 -0.09 -0.07 -0.16 -0.23 -0.06 -0.08 0.21 0.18 0.16
failures 0.07 0.28 0.06 0.04 0.09 -0.15 1.00 0.00 -0.03 -0.03 -0.28 -0.07 0.08 -0.05 0.10 0.07 0.12 0.11 0.05 0.10 -0.37 -0.38 -0.38
schoolsup -0.12 -0.20 -0.02 0.05 -0.03 0.07 0.00 1.00 0.09 0.00 0.08 -0.02 -0.09 -0.01 -0.03 -0.05 -0.03 -0.09 0.00 -0.01 -0.14 -0.09 -0.08
famsup -0.14 -0.12 -0.01 0.07 -0.03 0.14 -0.03 0.09 1.00 -0.01 0.09 0.08 -0.01 0.00 0.01 0.01 -0.02 -0.07 0.02 0.03 -0.01 -0.00 0.01
activities 0.12 -0.07 0.02 0.01 -0.03 0.08 -0.03 0.00 -0.01 1.00 0.06 0.07 0.04 0.05 0.13 0.07 -0.01 0.01 0.02 -0.01 0.07 0.06 0.03
higher -0.08 -0.24 -0.07 -0.00 -0.08 0.19 -0.28 0.08 0.09 0.06 1.00 0.06 -0.10 0.04 -0.09 -0.06 -0.11 -0.09 0.01 -0.07 0.27 0.25 0.24
internet 0.06 -0.03 -0.19 -0.01 -0.17 0.05 -0.07 -0.02 0.08 0.07 0.06 1.00 0.05 0.07 0.06 0.08 0.04 0.04 -0.04 0.09 0.10 0.12 0.11
romantic -0.11 0.17 0.02 0.01 0.01 0.04 0.08 -0.09 -0.01 0.04 -0.10 0.05 1.00 -0.05 0.01 0.00 0.05 -0.02 -0.00 0.11 -0.06 -0.10 -0.10
famrel 0.07 0.01 0.02 0.01 -0.01 0.01 -0.05 -0.01 0.00 0.05 0.04 0.07 -0.05 1.00 0.14 0.08 -0.08 -0.10 0.10 -0.06 0.04 0.04 0.05
freetime 0.18 0.00 0.01 0.01 -0.01 -0.09 0.10 -0.03 0.01 0.13 -0.09 0.06 0.01 0.14 1.00 0.32 0.14 0.13 0.08 -0.03 -0.05 -0.07 -0.06
goout 0.06 0.12 -0.03 -0.01 0.05 -0.07 0.07 -0.05 0.01 0.07 -0.06 0.08 0.00 0.08 0.32 1.00 0.25 0.40 -0.01 0.06 -0.10 -0.11 -0.10
Dalc 0.28 0.13 0.06 -0.08 0.11 -0.16 0.12 -0.03 -0.02 -0.01 -0.11 0.04 0.05 -0.08 0.14 0.25 1.00 0.63 0.07 0.13 -0.15 -0.13 -0.13
Walc 0.30 0.10 0.04 -0.09 0.08 -0.23 0.11 -0.09 -0.07 0.01 -0.09 0.04 -0.02 -0.10 0.13 0.40 0.63 1.00 0.11 0.14 -0.14 -0.13 -0.12
health 0.14 -0.03 0.01 0.01 -0.03 -0.06 0.05 0.00 0.02 0.02 0.01 -0.04 -0.00 0.10 0.08 -0.01 0.07 0.11 1.00 -0.03 -0.06 -0.09 -0.08
absences -0.01 0.15 -0.04 -0.02 -0.02 -0.08 0.10 -0.01 0.03 -0.01 -0.07 0.09 0.11 -0.06 -0.03 0.06 0.13 0.14 -0.03 1.00 -0.09 -0.09 -0.05
G1 -0.03 -0.12 -0.11 -0.06 -0.12 0.21 -0.37 -0.14 -0.01 0.07 0.27 0.10 -0.06 0.04 -0.05 -0.10 -0.15 -0.14 -0.06 -0.09 1.00 0.86 0.81
G2 -0.03 -0.12 -0.13 -0.06 -0.14 0.18 -0.38 -0.09 -0.00 0.06 0.25 0.12 -0.10 0.04 -0.07 -0.11 -0.13 -0.13 -0.09 -0.09 0.86 1.00 0.91
G3 -0.03 -0.13 -0.12 -0.06 -0.10 0.16 -0.38 -0.08 0.01 0.03 0.24 0.11 -0.10 0.05 -0.06 -0.10 -0.13 -0.12 -0.08 -0.05 0.81 0.91 1.00
sex age address famsize traveltime studytime failures schoolsup famsup activities higher internet romantic famrel freetime goout Dalc Walc health absences G1 G2 G3
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

Final Columns

1. school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2. sex - student's sex (binary: 'F' - female or 'M' - male)
3. age - student's age (numeric: from 15 to 22)
4. address - home address type (binary: 'U' - urban or 'R' - rural)
5. famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6. Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7. Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
8. Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9. Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10. Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
11. reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12. guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13. traveltime - school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14. studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15. failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16. schoolsup - extra educational support (binary: yes or no)
17. famsup - family educational support (binary: yes or no)
18. paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19. activities - extra-curricular activities (binary: yes or no)
21. higher - wants to take higher education (binary: yes or no)
22. internet - Internet access at home (binary: yes or no)
23. romantic - with a romantic relationship (binary: yes or no)
26. goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27. Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28. Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
1. G1 - first period grade (numeric: from 0 to 20)
2. G2 - second period grade (numeric: from 0 to 20)
3. G3 - final grade (numeric: from 0 to 20, output target)

: 나이

: 거주 지역

: 통학 시간

: 공부 시간

: F 받은 과목 수

: 대학원 진학 여부

: 인터넷 연결 유무

: 연애 여부

: 외출 횟수 (놀러)

: 하루 알코올 소비량

: 주간 알코올 소비량

: 이전 학기 성적

: 이전 학기 성적

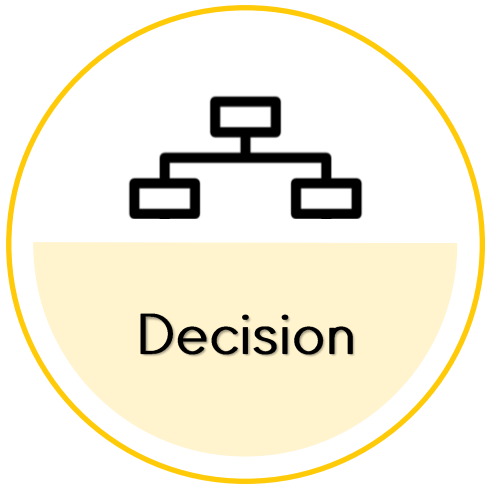
: 최종 성적 (target)

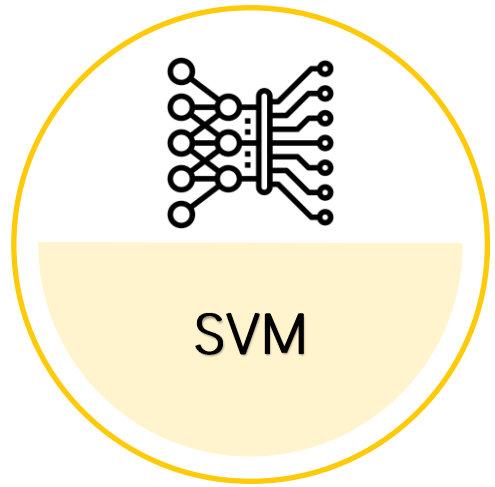
Separate Train set & Test set

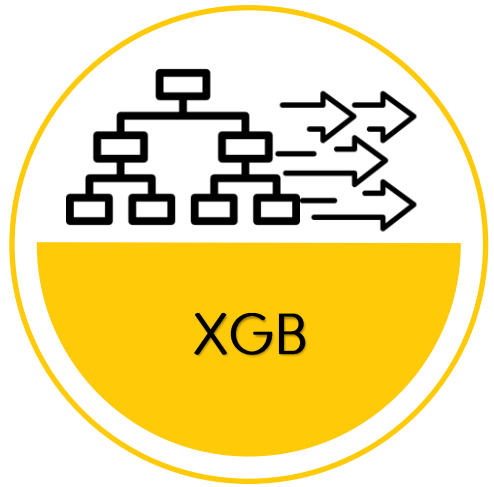
훈련세트 && 테스트 세트 나눠주기

```
from sklearn.model_selection import train_test_split

# data, target 정해 주기
data = studentACU1[['age', 'address', 'traveltime', 'studytime', 'failures', 'higher','internet', 'romantic','goout', 'Dalc', 'Walc', 'G1', 'G2']].to_numpy()

target = studentACU1['G3'].to_numpy()

# 훈련 & 테스트 나누기
train_input, test_input, train_target, test_target = train_test_split(data, target, test_size=0.3, shuffle=True)#, stratify=target, random_state=42)
```

Model for Project







Logistic regression

1번모델 : 로지스틱 회귀

```
# Logistic regression

from sklearn.linear_model import LogisticRegression

lgclassifier = LogisticRegression(solver = 'saga',random_state = 0)
lgclassifier.fit(train_input, train_target)
# y_pred = lgclassifier.predict(test_input)

a = lgclassifier.score(train_input, train_target)
b = lgclassifier.score(test_input, test_target)

print(lgclassifier.score(train_input, train_target))
print(lgclassifier.score(test_input, test_target))

# 결과 result 테이블에 넣기
result = result.append(pd.Series({'Model':'Logistic Regression','Train Accuracy':a,'Test Accuracy':b}),ignore_index=True )


# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

# 실행시 뜨는 aappend 에러 무시해도 괜찮음
```

```
C:\anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning: The max_iter was reached which means the c
  warnings.warn(

0.39863013698630134
0.3248407643312102
```

DecisionTree Regressor

2번 모델 : 결정 트리 Regressor

```
# DecisionTreeRegressor

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score

#하이퍼 파라미터 튜닝 X
# dt_r = DecisionTreeRegressor()

#하이퍼 파라미터 튜닝 후
dt_r =  DecisionTreeRegressor(
        max_depth=4,
        criterion='squared_error',
        min_samples_leaf=13
    )
dt_r.fit(train_input, train_target)

a = dt_r.score(train_input, train_target)
b = dt_r.score(test_input, test_target)

print(dt_r.score(train_input, train_target))
print(dt_r.score(test_input, test_target))

# 결과 result 테이블에 정확도 넣기
result = result.append(pd.Series({'Model':'Decison Tree Regressor','Train Accuracy':a,'Test Accuracy':b}),ignore_index=True )

#파라미터 튜닝 전
# 0.9996779659649728
# 0.5814954475824354

# 튜닝 후
# 0.8648312632709616
# 0.7484985252931828

# https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
```

```
0.8344589913147038
0.8273303642098812
```

DTR - Hyper parameter tuning

결정트리 Regressor - 하이퍼 파라미터 튜닝

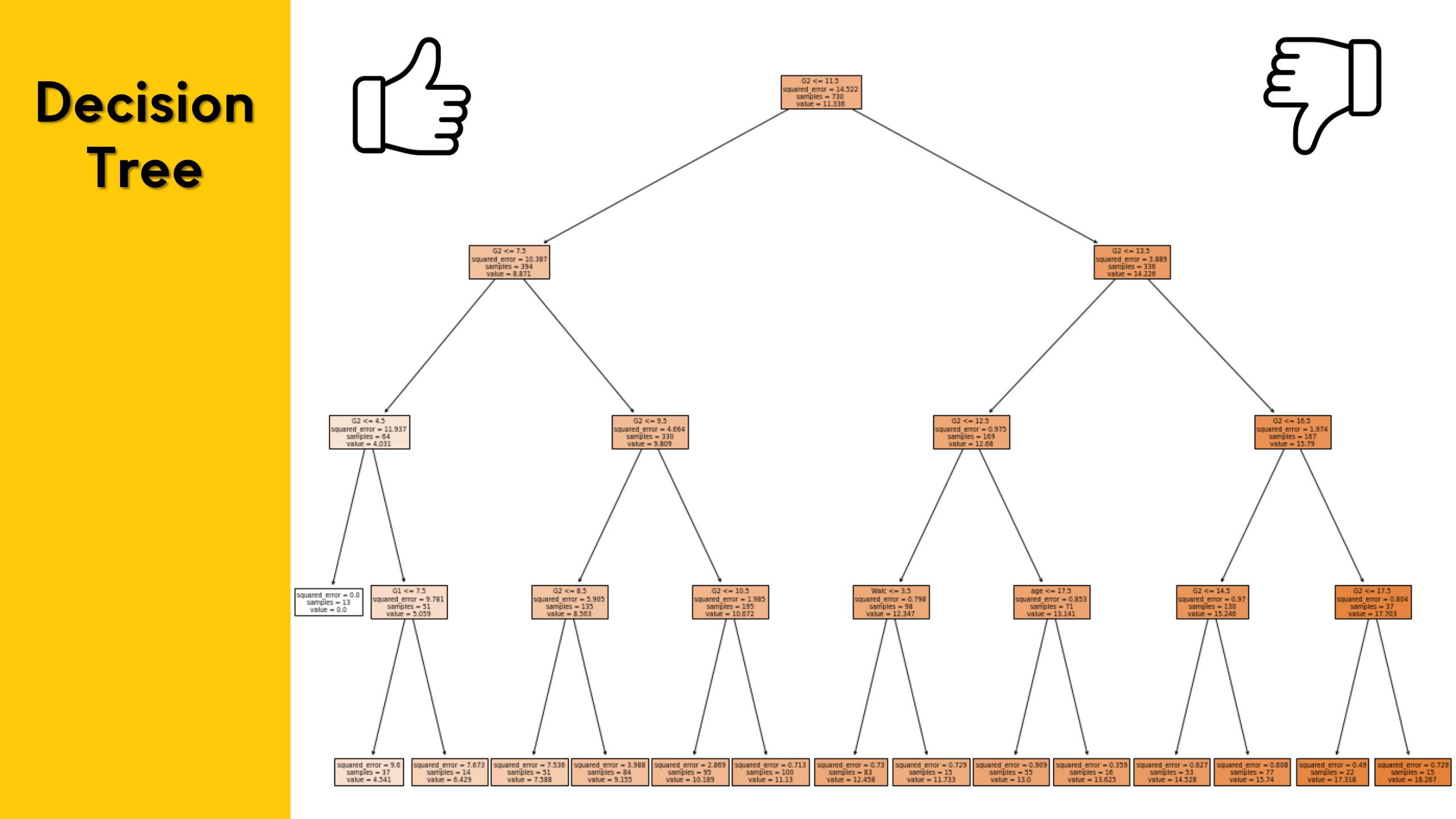
```
#RandomizedSearchCV를 사용하여 파라미터값을 랜덤 대입해 최적의 파라미터 값을 검색.
from scipy.stats import randint
from sklearn.model_selection import RandomizedSearchCV

# DecisionTree Regressor 파라미터 값 설명.
#
# max_depth
#


DTRG_params ={
                'criterion':['squared_error','friedman_mse'],
                'max_depth':randint(low=1, high=10),
                # 'max_leaf_nodes':[None,2,3,4,5,6,7],
                # 'min_samples_split':[2,3,4,5,6],
                'min_samples_leaf': randint(low=10, high=50),
                # 'max_features':[None,'sqrt','log2',3,4,5]
            }

DCRG_RD = DecisionTreeRegressor()
RDSearch = RandomizedSearchCV(DCRG_RD, param_distributions=DTRG_params, n_iter=50, cv=5, scoring='neg_mean_squared_error')
RDSearch.fit(train_input, train_target)

CV_result = RDSearch.cv_results_

#결과 값 오차순으로 정렬하여, 가장 오차가 낮은 파라미터를 확인 후, 위 모델에 다시 대입.
tunning_result = list(zip(CV_result["mean_test_score"], CV_result["params"]))
tunning_result.sort(reverse=True, key=lambda x : x[0])

for mean_score, params in tunning_result:
    print(np.sqrt(-mean_score), params)
```

Make Decision Tree

결정트리 Regressor - 결정트리 그리기

```
# matplotlib 로 결정트리 그려보기

import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(20,15))
plot_tree(dt_r, filled=True, feature_names=['age', 'address', 'traveltime', 'studytime'\
    'failures', 'higher','internet', 'romantic','goout', 'Dalc', 'Walc', 'G1', 'G2'])

plt.savefig("dt_r_result.pdf") #저장
plt.show()
```

Decision Tree



SVM

3번 모델: 서프트 벡터 머신 Kernelized

```
from sklearn import svm

KSVC_clf = svm.SVC(kernel='sigmoid',C=10,gamma=0.001)
KSVC_clf.fit(train_input, train_target)

a = KSVC_clf.score(train_input, train_target)
b = KSVC_clf.score(test_input, test_target)

print(KSVC_clf.score(train_input, train_target))
print(KSVC_clf.score(test_input, test_target))

# 결과 result 테이블에 정확도 넣기
result = result.append(pd.Series({'Model':'Kernelized SVM','Train Accuracy':a,'Test Accuracy':b}),ignore_index=True )


# https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
```

```
0.3643835616438356
0.34394904458598724
```

Random Forest Regressor

4번 모델 : 랜덤 포레스트 regressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression

# 'max_depth': 5, 'min_samples_leaf': 8, 'min_samples_split': 16, 'n_estimators': 100
# regr = RandomForestRegressor(max_depth=5, min_samples_leaf=8, min_samples_split=16,n_estimators=100)#random_state=0)
#regr = RandomForestRegressor() #하이퍼 파라미터 튜닝 X

#하이퍼 파라미터 튜닝 완료 값.
regr = RandomForestRegressor(max_depth=10,max_features=7,n_estimators=173)



regr.fit(train_input, train_target)

a = regr.score(train_input, train_target)
b = regr.score(test_input, test_target)


print(regr.score(train_input, train_target))
print(regr.score(test_input, test_target))

# 결과 result 테이블에 정확도 넣기
result = result.append(pd.Series({'Model':'RandomForestRegressor','Train Accuracy':a,'Test Accuracy':b}),ignore_index=True )

# 튜닝 전
# 0.9767455733333666
# 0.7403084516586431

# 튜닝 후
# 0.9685004436756117
# 0.756712722032685
```

```
0.9618267725730998
0.8105195081951718
```

RFR - Hyper parameter tuning

랜덤포레스트 Regressor - 하이퍼 파라미터 튜닝

+ 코드 + Markdown

```
#RandomizedSearchCV를 사용하여 파라미터값을 랜덤 대입해 최적의 파라미터 값을 검색.
from scipy.stats import randint
from sklearn.model_selection import RandomizedSearchCV

# RandomForestRegressor 파라미터 값 설명.
# n_estimators: 생성할 Tree 개수
# criterion : 분할 품질을 측정하는 기능 (default : gini)
# max_depth : 트리의 최대 깊이
# max_features : 각 노드에서 분할에 사용할 특징의 최대 수


RFRG_params ={
    'n_estimators': randint(low=1, high=200),
    'max_features': randint(low=1, high=8),
    'max_depth' : randint(low=1, high = 50)
}

RFRG_RD = RandomForestRegressor()
RDSearch = RandomizedSearchCV(RFRG_RD, param_distributions=RFRG_params, n_iter=50, cv=5, scoring='neg_mean_squared_error')
RDSearch.fit(train_input, train_target)

CV_result = RDSearch.cv_results_

#결과 값 오차순으로 정렬하여, 가장 오차가 낮은 파라미터를 확인 후, 위 모델에 다시 대입.
tunning_result = list(zip(CV_result["mean_test_score"], CV_result["params"]))
tunning_result.sort(reverse=True)

for mean_score, params in tunning_result:
    print(np.sqrt(-mean_score), params)
```

XGB Regressor

5번 모델 : XGBRegressor

```
from xgboost import XGBRegressor
from sklearn.model_selection import cross_validate

#하이퍼파라미터 튜닝X
# xgb = XGBRegressor()

# xgb = XGBRegressor(max_depth=2, min_samples_leaf=8, min_sample_split=16, n_estimators=100)#tree_method='hist', random_state=42)

#dart 방식은 과적합이 상대적으로 심하여 gbtree 방식 booster 선택.
# xgb = XGBRegressor(skip_drop=0, sample_type= 'uniform', rate_drop= 0.2, booster='dart')
xgb = XGBRegressor(booster ='gbtree', learning_rate=0.1,max_depth=3,n_estimators=63)


scores = cross_validate(xgb, train_input, train_target, return_train_score=True, n_jobs=-1)

a = np.mean(scores['train_score'])
b = np.mean(scores['test_score'])

print(np.mean(scores['train_score']))
print(np.mean(scores['test_score']))

# 결과 result 테이블에 정확도 넣기
result = result.append(pd.Series({'Model':'XGBRegressor','Train Accuracy':a,'Test Accuracy':b}),ignore_index=True )

#하이퍼파라미터 튜닝X
# 0.9976330068351025
# 0.8052925704563478

#하이퍼파라미터 튜닝 후
# 0.9067115016713991
# 0.8434975104140227
# print(f"Train score: {np.mean(scores['train_score'])}, Test score: {np.mean(scores['test_score'])}")
```

```
0.8792972839980063
0.800289055234229
```

XGBR - Hyper parameter tuning

XGBRegressor - 하이퍼 파라미터 튜닝

```
#RandomizedSearchCV를 사용하여 파라미터값을 랜덤 대입해 최적의 파라미터 값을 검색.
from scipy.stats import randint
from sklearn.model_selection import RandomizedSearchCV

# XGBRegressor 파라미터 값 설명.
# booster : 어떤 부스터 구조를 쓸지 결정(gbtree, gblinear, dart)
# eta: learning rate. 트리에 가지가 많을 수록 과적합. 매 부스팅 스탭마다 weight를 주어 부스팅 과정에 과적합이 일어나지 않도록 한다.
# gamma: 정보획득(information Gain)에서 -r로 표현한 바 있다. 이것이 커지면, 트리 깊이가 줄어들어 보수적인 모델이 된다. ( 디폴트는 0 )
# max_depth : 한 트리의 maxium depth.
# lambda (L2 reg-form) : L2 Regularization Form에 달리는 weights. 숫자가 클수록 보수적인 모델.
# alpha(L1 reg-form) : L1 Regularization Form에 달리는 weights. 숫자가 클수록 보수적인 모델.
# objective : 목적함수이다. reg:linear(linear-regression), binary:logistic(binary-logistic-classification), count:poisson(count data poison regression) 등 다양
# eval_metric : 모델의 평가 함수를 조정하는 함수 - rmse(root mean square error), logloss(log-likelihood), map(mean average precision) 등 데이터의 특성에 맞게 평가 함수
# num_rounds : 부스팅 라운드를 결정한다. 랜덤하게 생성되는 모델이니만큼 이 수 가 적당히 큰게 좋다 epochs 옵션과 동일하다 .

#param_distributions를 아래 두 방식중 선택하여 사용
#gbtree 방식의 booster가 과적합이 덜하여 해당 방식으로 선택.
XGBR_params ={
    'booster' : ['gbtree'], #선형모델은 목적과 맞지않으니 제외
     'n_estimators' : randint(low=50, high=1000),
     'learning_rate' : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
     'max_depth': randint(low=2, high=15)
}
XGBDart_params ={
    'booster' : ['dart'],
    'sample_type' : ['uniform', 'weighted'],
    'rate_drop' : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
    'skip_drop' : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
}

XGBR_RD = XGBRegressor()
RDSearch = RandomizedSearchCV(XGBR_RD, param_distributions=XGBR_params, n_iter=50, cv=5, scoring='neg_mean_squared_error')
RDSearch.fit(train_input, train_target)

CV_result = RDSearch.cv_results_

#결과 값 오차순으로 정렬하여, 가장 오차가 낮은 파라미터를 확인 후, 위 모델에 다시 대입.
tunning_result = list(zip(CV_result["mean_test_score"], CV_result["params"]))
tunning_result.sort(reverse=True, key=lambda x : x[0])

for mean_score, params in tunning_result:
    print(np.sqrt(-mean_score), params)
```

Final Score

모델 하이퍼 파라미터 테이블 (모델별 최종 스코어)

```
# result = pd.DataFrame(columns=['Model','Train Accuracy','Test Accuracy']) # 초기화
result.sort_values(by='Test Accuracy', ascending=False)
```

	Model	Train Accuracy	Test Accuracy
1	Decison Tree Regressor	0.834459	0.82733
3	RandomForestRegressor	0.961827	0.81052
4	XGBRegressor	0.879297	0.800289
2	Kernelized SVM	0.364384	0.343949
0	Logistic Regression	0.39863	0.324841

With Score

1. 결정트리 Regressor

[49]:
```
# DecisionTreeRegreossor

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score


#하이퍼 파라미터 튜닝 후
dt_r =  DecisionTreeRegressor(
        max_depth=4,
        criterion='squared_error',
        min_samples_leaf=13
    )
dt_r.fit(train_input, train_target)

a = dt_r.score(train_input, train_target)
b = dt_r.score(test_input, test_target)

print(dt_r.score(train_input, train_target))
print(dt_r.score(test_input, test_target))

# 예측
print(dt_r.predict([data_scoreInclude[0, :]]))
# print(dt_r.predict([data_scoreExcept[0, :]]))
```

```
0.8317798799269167
0.8301097087229754
[4.56521739]
```

Without Score

1. 결정트리 Regressor

[90]:
```
# DecisionTreeRegreossor

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score


#하이퍼 파라미터 튜닝 후
dt_r =  DecisionTreeRegressor(
        max_depth=4,
        criterion='squared_error',
        min_samples_leaf=13
    )
dt_r.fit(train_input, train_target)

a = dt_r.score(train_input, train_target)
b = dt_r.score(test_input, test_target)

print(dt_r.score(train_input, train_target))
print(dt_r.score(test_input, test_target))

# 예측
# print(dt_r.predict([data_scoreInclude[0, :]]))
print(dt_r.predict([data_scoreExcept[0, :]]))
```

```
0.24182403739668956
0.12483824275369193
[12.26234568]
```

With Score

2. 랜덤포레스트 Regressor

[50]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression


regr = RandomForestRegressor(max_depth=10,max_features=7,n_estimators=173)


regr.fit(train_input, train_target)

a = regr.score(train_input, train_target)
b = regr.score(test_input, test_target)


print(regr.score(train_input, train_target))
print(regr.score(test_input, test_target))


# 예측
print(regr.predict([data_scoreInclude[0, :]]))
# print(regr.predict([data_scoreExcept[0, :]]))
```

```
0.9606826877813853
0.8068728115102017
[5.95947564]
```

Without Score

2. 랜덤포레스트 Regressor

[91]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression


regr = RandomForestRegressor(max_depth=10)#,max_features=7,n_estimators=173)


regr.fit(train_input, train_target)

a = regr.score(train_input, train_target)
b = regr.score(test_input, test_target)


print(regr.score(train_input, train_target))
print(regr.score(test_input, test_target))


# 예측
# print(regr.predict([data_scoreInclude[0, :]]))
print(regr.predict([data_scoreExcept[0, :]]))
```

```
0.6491316233113951
0.135381041822831 28
[10.0758334]
```

With Score

3. XGBRegressor

[51]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import cross_validate


xgb = XGBRegressor(booster ='gbtree', learning_rate=0.1,max_depth=3,n_estimators=
scores = cross_validate(xgb, train_input, train_target, return_train_score=True,

xgb.fit(train_input, train_target)

a = np.mean(scores['train_score'])
b = np.mean(scores['test_score'])

print(np.mean(scores['train_score']))
print(np.mean(scores['test_score']))


# 예측
print(xgb.predict([data_scoreInclude[0, :]]))
# print(xgb.predict([data_scoreExcept[0, :]]))
```

```
0.8839850228899764
0.829043284718695
[6.0279393]
```

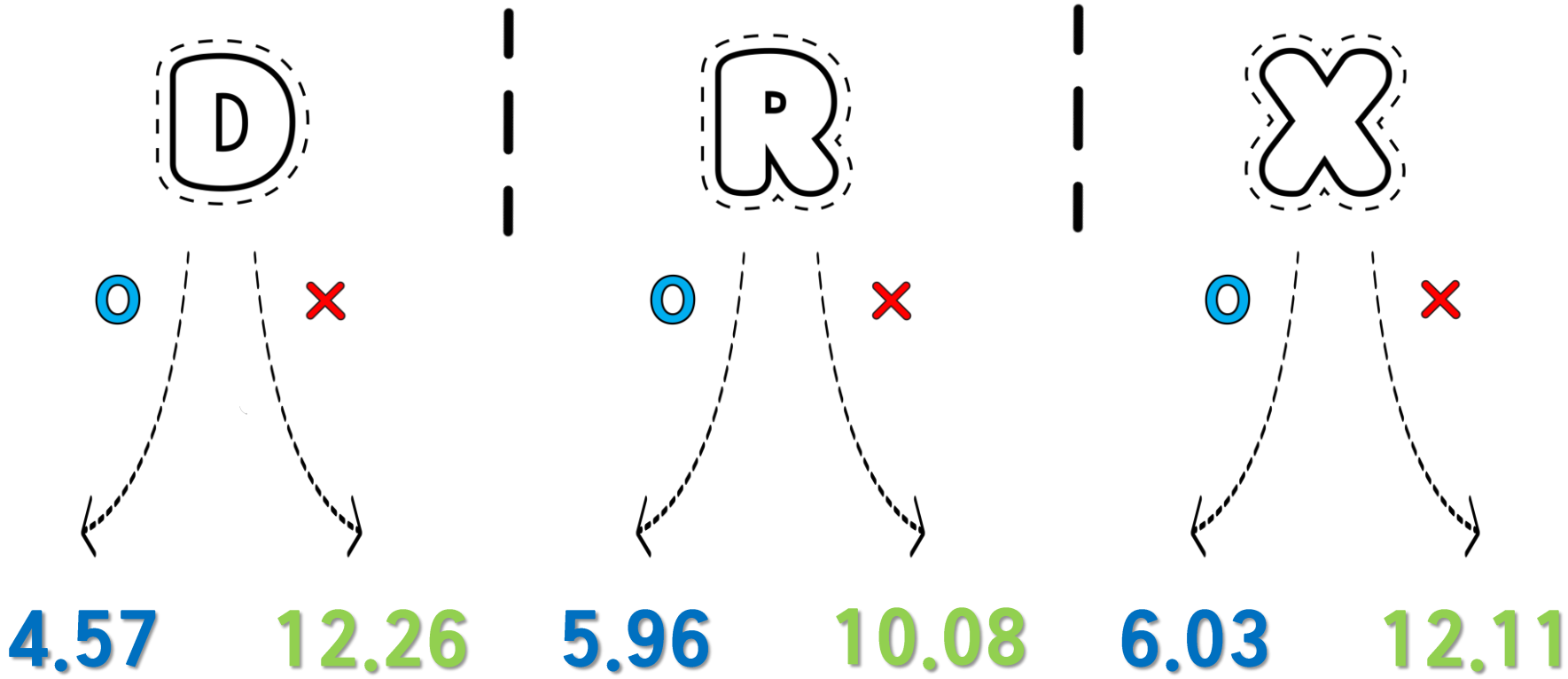
Without Score

3. XGBRegressor

[92]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import cross_validate


xgb = XGBRegressor(booster ='gbtree', learning_rate=0.1,max_depth=3,n_estimators=
scores = cross_validate(xgb, train_input, train_target, return_train_score=True,

xgb.fit(train_input, train_target)

a = np.mean(scores['train_score'])
b = np.mean(scores['test_score'])

print(np.mean(scores['train_score']))
print(np.mean(scores['test_score']))


# 예측
# print(xgb.predict([data_scoreInclude[0, :]]))
print(xgb.predict([data_scoreExcept[0, :]]))
```

```
0.379424952508930 93
0.1687229230620157
[12.108138]
```

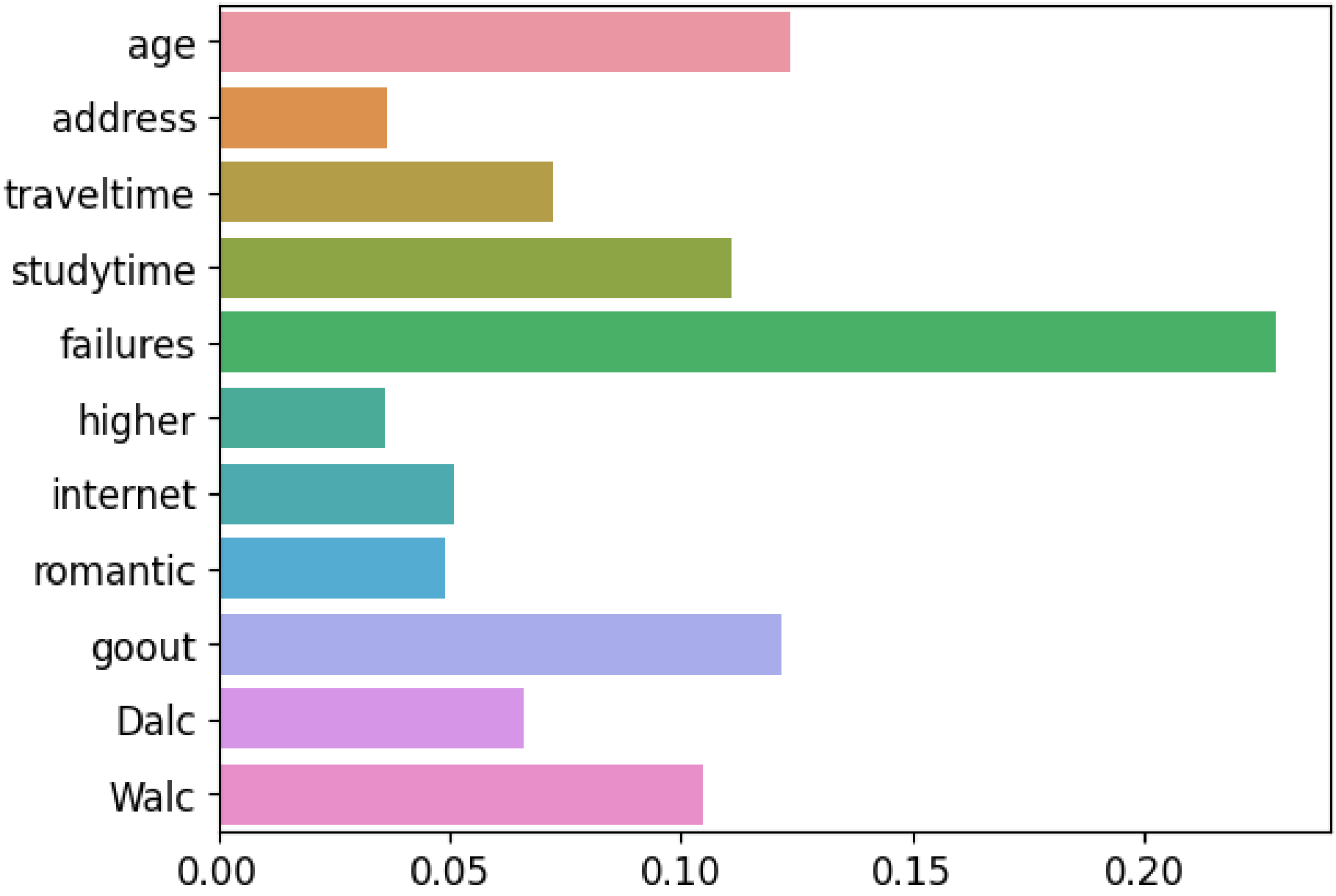
Final Score



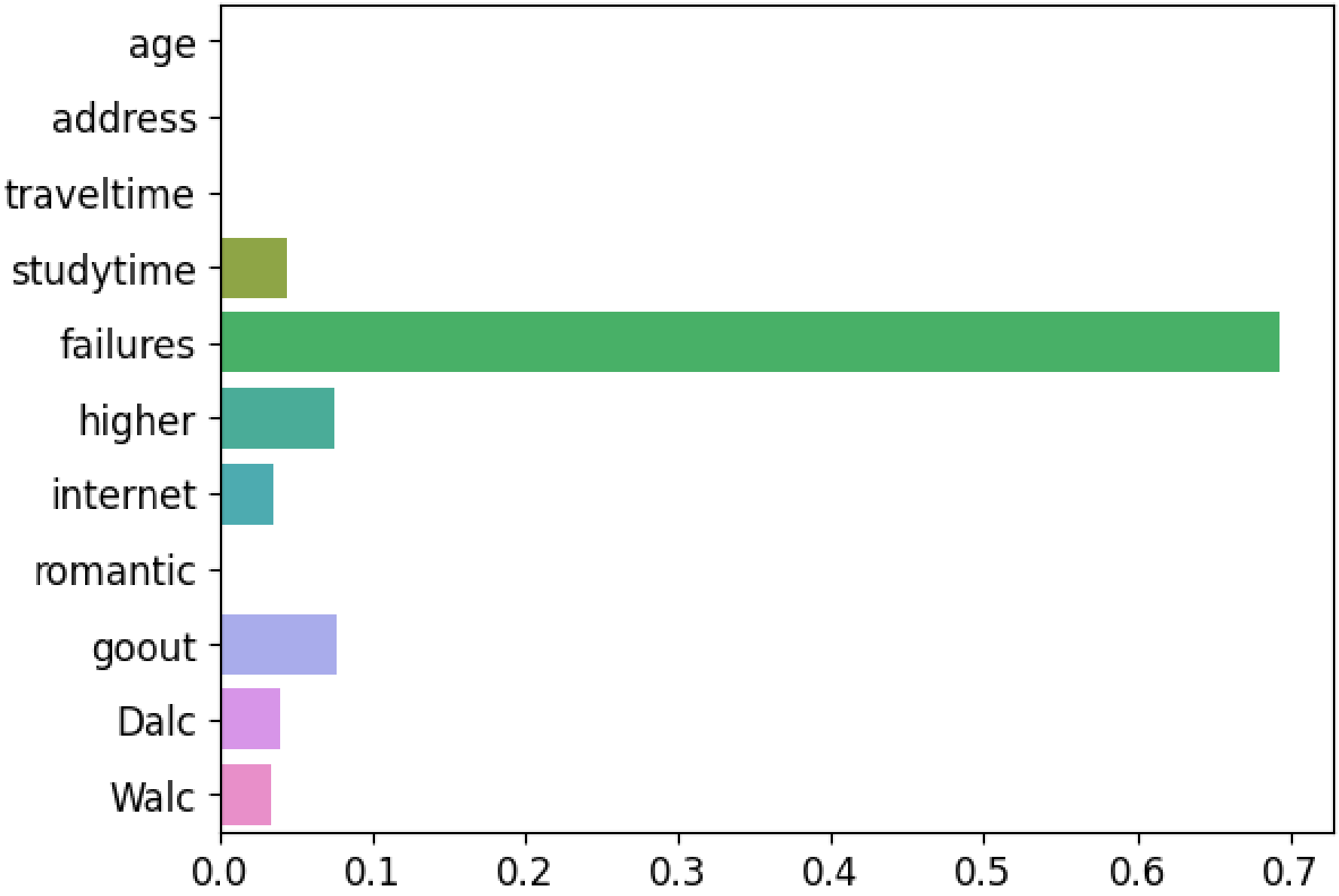
Exist score: O

Not exist score: X

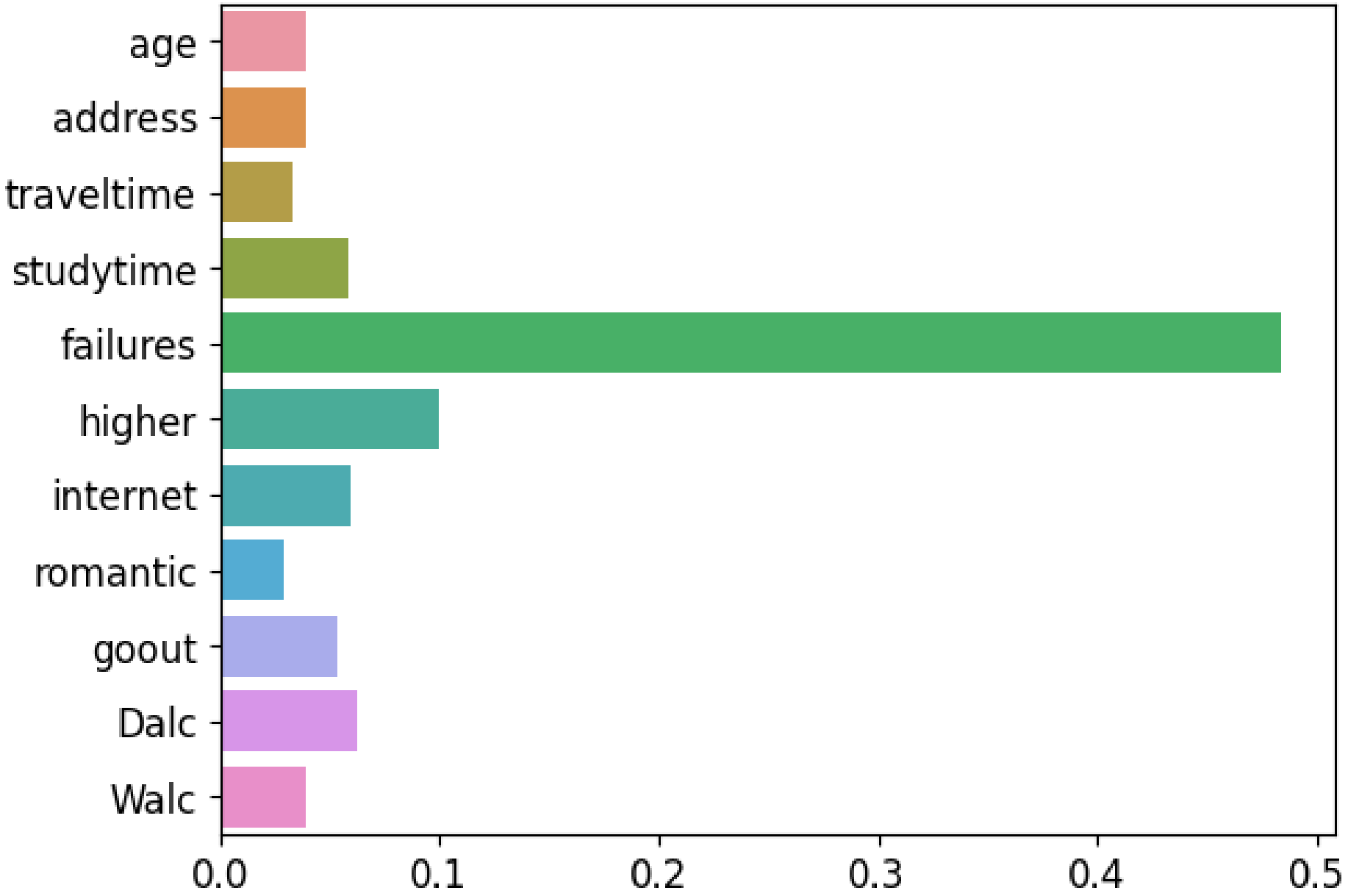
Random Forest Feature Importance



Decision Tree Feature Importance



XGB Feature Importance



Project Result

- With G1, G2

```
1. What is student's age?(15~22)
A) 22
2. Where is the student's address, urban(0) or rural(1)?
A) 0
3. How long does the average student traveltime to school?(1~4)
A) 3
4. What is the student's weekly study time?(1~4)
A) 2
5. What is the number of past class failures for a student?(0~3)
A) 0
6. Does the student want to get a higher education?(0,1)
A) 0
7. Is there an internet connection at the student's house?(0,1)
A) 1
8. Is the student in a romantic relationship?(0,1)
A) 0
9. How often does a student go out with his friends?(1~5)
A) 2
10. What is the student's daily alcohol consumption?(1~5)
A) 1
11. What is the student's weekend alcohol consumption?(1~5)
A) 1
12. What is the grade of a student in the first semester?(0~20)
A) 15
13. What is the grade of a student in the second semester?(0~20)
A) 15
Please select a model(1:dtr, 2:regr, 3:xgb):  3
Please enter your name to check your prediction score:  hyeonjung
The perfect score is 20.
hyeonjung's prediction score for the last semester is [15.281355].
```

Project Result

- With G1, G2

```
1. What is student's age?(15~22)
A) 22
2. Where is the student's address, urban(0) or rural(1)?
A) 0
3. How long does the average student traveltime to school?(1~4)
A) 1
4. What is the student's weekly study time?(1~4)
A) 2
5. What is the number of past class failures for a student?(0~3)
A) 0
6. Does the student want to get a higher education?(0,1)
A) 0
7. Is there an internet connection at the student's house?(0,1)
A) 1
8. Is the student in a romantic relationship?(0,1)
A) 1
9. How often does a student go out with his friends?(1~5)
A) 3
10. What is the student's daily alcohol consumption?(1~5)
A) 4
11. What is the student's weekend alcohol consumption?(1~5)
A) 5
12. What is the grade of a student in the first semester?(0~20)
A) 15
13. What is the grade of a student in the second semester?(0~20)
A) 16
Please select a model(1:dtr, 2:regr, 3:xgb):  2
Please enter your name to check your prediction score:  HyunSoo
The perfect score is 20.
HyunSoo's prediction score for the last semester is [16.28583333].
```

Project Result

- Without G1, G2

```
1. What is student's age?(15~22)
A) 22
2. Where is the student's address, urban(0) or rural(1)?
A) 0
3. How long does the average student traveltime to school?(1~4)
A) 1
4. What is the student's weekly study time?(1~4)
A) 2
5. What is the number of past class failures for a student?(0~3)
A) 0
6. Does the student want to get a higher education?(0,1)
A) 0
7. Is there an internet connection at the student's house?(0,1)
A) 1
8. Is the student in a romantic relationship?(0,1)
A) 1
9. How often does a student go out with his friends?(1~5)
A) 1
10. What is the student's daily alcohol consumption?(1~5)
A) 1
11. What is the student's weekend alcohol consumption?(1~5)
A) 1
Please select a model(1:dtr, 2:regr, 3:xgb):  1

==============================SCORE==============================
Please enter your name to check your prediction score:  혜성
혜성's prediction score for the last semester is [9.86666667].
The perfect score is 20.
=================================================================
```

Project Result

- Without G1, G2

```
1. What is student's age?(15~22)
A) 22
2. Where is the student's address, urban(0) or rural(1)?
A) 0
3. How long does the average student traveltime to school?(1~4)
A) 1
4. What is the student's weekly study time?(1~4)
A) 2
5. What is the number of past class failures for a student?(0~3)
A) 0
6. Does the student want to get a higher education?(0,1)
A) 0
7. Is there an internet connection at the student's house?(0,1)
A) 1
8. Is the student in a romantic relationship?(0,1)
A) 1
9. How often does a student go out with his friends?(1~5)
A) 3
10. What is the student's daily alcohol consumption?(1~5)
A) 1
11. What is the student's weekend alcohol consumption?(1~5)
A) 2
Please select a model(1:dtr, 2:regr, 3:xgb):  2

==============================SCORE==============================
Please enter your name to check your prediction score:  gaeun
gaeun's prediction score for the last semester is [10.80658517].
The perfect score is 20.
=================================================================
```

Conclusion

“음주가 공부에 큰 영향을 미칠 것이다”

But, 알코올은 성적에 큰 영향 x, 직전 학기 성적이 가장 연관

성적에 있어 환경적 요인 큰 영향 x

환경적인 요소보다 학업 역량이 큰 영향

성적에 가장 큰 영향을 미치는 요소: **기존 학업 역량**

아쉬운 점:

너무 적은 데이터

각 성적마다의 **환경 변화 요소**가 있었다면?

좋았던 점:

여러 모델을 사용해 볼 수 있었음

나름 **일정한 결과**를 도출할 수 있었음


Thank You

Any Question?