Student’s Mental Health & Productivity

회귀 모델 구현 및 분석

**과목 : 기계학습**

**학번 : 202020827**

**이름 : 김경민**

**제출일 : 2025-04-23**

**📌 목차**

**1. 데이터셋 설명**

* 1-1. 프로젝트 개요
* 1-2. 데이터셋 설명

**2. 데이터 전처리 및 탐색적 분석 (EDA)**

* 2-1. 범주형 변수처리
* 2-2. 결측치 처리
* 2-3. 스케일링
* 2-4. EDA 시각화

**3. 모델 구축 및 학습**

* 3-1. 사용한 알고리즘
* 3-2. 데이터 분할 방식
* 3-3. 파이프라인 구성

**4. 성능 평가**

* 4-1. 사용한 지표
* 4-2. 예측 결과 시각화
* 4-3. 모델별 평가 요약
* 4-4. 해석

**5. 하이퍼파라미터 튜닝**

* 5-1. 튜닝 방법
* 5-2. 튜닝한 하이퍼파라미터 & CV 성능
* 5-3. 튜닝 결과 분석 및 파라미터 해석

**6. 결론 및 고찰**

* 6-1. 최종 모델 성능 종합 평가
* 6-2. 데이터 및 모델의 한계
* 6-3. 실생활 응용 가능성 및 확장 방향
* 6-4. 다음 단계에서 고려할 점

**7. 데이터셋 출처 및 참고 자료**

* 7-1. 공식 문서
* 7-2. 참고 블로그
* 7-3. 사용한 주요 라이브러리

**8. 부록**

**📌 1.데이터셋 설명**

**1-1. 프로젝트 개요**  
문제 정의  
학생 설문·인구통계학·정신 건강 지표를 통합하여 CGPA\_numeric(0.00–4.00) 연속값을 예측하는 회귀 문제

목표

* 설문 응답 구간(“0–1.99” … “3.50–4.00”)을 중앙값으로 변환한 실수형 타깃(CGPA\_numeric) 예측 모델 개발
* RMSE·MAE·R² 등 다채널 평가 지표로 성능 검증
* GridSearchCV/RandomizedSearchCV로 최적 하이퍼파라미터 탐색

기대 효과

* 정신 건강 요인이 학업 성취도에 미치는 정량적 인사이트 제공
* 조기 위험 학생 선별 및 맞춤형 학습·상담 프로그램 설계 근거 마련
* 교육 기관의 데이터 기반 학생 관리·상담 체계 강화

**1-2. 데이터셋 설명**  
출처  
Kaggle – Students’ Mental Health & Productivity  
<https://www.kaggle.com/datasets/shariful07/student-mental-health>

데이터 규모

* 샘플 수: 102개
* 변수 수: 11개 (연속형 2, 순서형 1, 범주형 3, 이진형 5)

종속변수(target)

* CGPA\_numeric: 설문 응답 구간을 중앙값으로 매핑한 실수형 연속 변수

독립변수(features)

* Choose your gender: 성별(범주형)
* Age: 만 나이(연속형)
* What is your course?: 전공명(범주형)
* Your current year of Study: 학년(Year 1→1 … Year 4→4, 순서형)
* Marital status: 결혼 여부(범주형)
* Do you have Depression?: 우울증 여부(이진형)
* Do you have Anxiety?: 불안 여부(이진형)
* Do you have Panic attack?: 공황 발작 경험 여부(이진형)
* Did you seek any specialist for a treatment?: 전문의 상담 이력(이진형)

**📌 2. 데이터 전처리 및 탐색적 분석(EDA)**

**2-1. 범주형 변수 처리**

* Ordinal 매핑 (학년)
  + Your current year of Study → 문자열 정제 → Year 1…Year 4 → 정수 1~4 매핑
  + 설명: 학년 정보는 순서형 데이터이므로, 모델이 학년 간 순서를 학습할 수 있도록 수치형으로 변환
* One-Hot Encoding
  + get\_dummies(..., drop\_first=True)
  + 대상: 성별, 전공, 이진 질문들(Depression, Anxiety 등)
  + 설명:
    - 범주형 변수를 0/1 형식의 더미 변수로 변환하여 회귀 모델에 투입
    - drop\_first=True 로 기준(dummy trap 방지)

# 학년 One-Hot Encoding

data['Study\_Year\_Str'] = data['Your current year of Study'].str.title()

year\_dummies = pd.get\_dummies(data['Study\_Year\_Str'], prefix='Year', drop\_first=False)

data = pd.concat([data, year\_dummies], axis=1)

# 기타 범주형 One-Hot

data = pd.get\_dummies(data,

columns=['Choose your gender','What is your course?'] + orig\_cat,

drop\_first=True)

**2-2. 결측치 처리**

* 수치형 변수(Age, Study\_Year, CGPA\_numeric)
  + 방법: SimpleImputer(strategy='mean')
  + 설명:
    - 평균 대체를 통해 극단치에 크게 영향을 받지 않으면서 전체 분포를 보존
    - CGPA\_numeric은 이미 구간→중앙값 매핑으로 변환된 연속형 변수이므로, 평균 대체 시 해석 의미 유지
* 범주형 변수(원본 설문 항목들)
  + 방법: SimpleImputer(strategy='most\_frequent')
  + 설명:
    - 결측치가 많은 문자열 변수(성별·전공·이진 질문)에 대해서는 최빈값으로 대체하여 가장 일반적인 범주로 채움
    - 카테고리 수가 많지 않아, 결측이 전체 분포에 미치는 왜곡 최소화

from sklearn.impute import SimpleImputer

# 수치형 결측치 대체

num\_cols = ['Age','Study\_Year','CGPA\_numeric']

data[num\_cols] = SimpleImputer(strategy='mean').fit\_transform(data[num\_cols])

# 범주형 결측치 대체

orig\_cat = ['Choose your gender','What is your course?',

'Do you have Depression?','Do you have Anxiety?',

'Do you have Panic attack?','Marital status',

'Did you seek any specialist for a treatment?']

imp = SimpleImputer(strategy='most\_frequent').fit\_transform(data[orig\_cat])

data[orig\_cat] = pd.DataFrame(imp, columns=orig\_cat, index=data.index)

**2-3. 스케일링**

* StandardScaler 적용
  + 대상: Age, Study\_Year, CGPA\_numeric
  + 설명:
    - 평균 0, 표준편차 1로 정규화하여 변수별 스케일 차이를 제거
    - 회귀 계수 해석 시 비교 가능하도록 단위 통일

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data[['Age','Study\_Year','CGPA\_numeric']] = scaler.fit\_transform(

data[['Age','Study\_Year','CGPA\_numeric']]

)

**2-4. EDA 시각화 및 해석**

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.스크린샷, 직사각형, 다채로움, 사각형이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

도표, 텍스트, 스크린샷, 라인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 라인, 그래프, 도표, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다..

2-4. EDA 시각화

**1) CGPA\_numeric 분포**

• 히스토그램 + KDE (이상치 제거 후)

- CGPA\_numeric 값이 –0.3에서 0.7 사이에 70% 이상 집중

- KDE 곡선이 한 봉우리를 형성하며 꼬리가 대칭에 가깝고 길지 않아 왜도(skew) 작음

• Shapiro–Wilk 검정

- p-value ≈ 0.07 (> 0.05) → 귀무가설(정규분포) 채택

- 샘플 분포가 크게 치우치지 않음을 통계적으로 확인

• Q–Q Plot

- 대부분 점이 이론치 직선 근처에 위치

- 양 끝 극단치 구간에서만 소폭 벗어남 → 대체로 정규성 유지

**2) 수치형 변수 히스토그램**

• Age Distribution

- Z-score 변환 후 –1~1 구간에 약 80% 데이터 집중 → 연령 편차 크지 않음

• Study\_Year Distribution

- 1학년·3학년이 각각 약 30%씩 차지, 2·4학년은 20% 내외 → 학년별 표본 불균형 존재

• CGPA\_numeric Distribution

- Z-score 변환 후 중앙(0) 부근 밀집 → 전체 CGPA 편차가 크지 않고 평균 근처에서 분포

**3) 수치형 변수 상관관계 (Heatmap)**

• Pearson 상관계수

- Age–Study\_Year: ρ≈0.23 (약한 양의 상관) → 학년이 올라갈수록 연령도 증가

- CGPA\_numeric–Age: ρ≈0.01, CGPA\_numeric–Study\_Year: ρ≈–0.00 → 선형 관계 거의 없음

• Spearman 상관계수

- 순위 기반 상관도 Pearson과 유사 → 비선형 관계나 순서 효과도 미미

4**) 이진 변수별 CGPA\_numeric 박스플롯**

• “우울증” 유무

- 두 그룹 메디안 차이 < 0.05, IQR 겹침 → 우울증 단일 변수로 CGPA 구분 어려움

• “불안”·“공황” 유무

- 중위수·사분범위 유사, 일부 아웃라이어만 소수 존재

• “결혼 여부”

- 전체 데이터의 95%가 미혼, 기혼 표본 부족으로 분포 비교 한계

• “진료 여부”

- 진료 받은 그룹 메디안이 약간 낮음(0.1 이하), 분포 폭은 유사 → 스트레스 지표로 일부 설명 가능

**📌 3. 모델 구축 및 학습**

**3-1. 사용한 알고리즘**  
• LinearRegression  
• Ridge (α=1.0)  
• Lasso (α=0.1)  
• DecisionTreeRegressor (max\_depth=5)

**3-2. 데이터 분할 방식**  
• 특징·타겟 분리  
– X ← clean[features], y ← clean['CGPA\_numeric'] 로 독립변수·종속변수 설정  
• train\_test\_split 사용  
– from sklearn.model\_selection import train\_test\_split  
– test\_size=0.2 → 학습용 80%·테스트용 20% 비율 설정  
– random\_state=42 → 난수 고정으로 실행할 때마다 동일한 결과 재현  
– shuffle=True 기본값 적용 → 데이터를 무작위로 섞은 뒤 분할하여 편향 방지  
– stratify=None (회귀 문제이므로 클래스 균형 목적의 stratify 미적용)

**3-3. 파이프라인 구성**  
• Pipeline 도입 배경  
– 전처리(스케일링)와 모델 학습 단계를 하나의 객체에 묶어 코드 간결화  
– .fit(), .predict() 만 호출하면 전처리→학습→예측이 자동 순차 실행  
– GridSearchCV 연동 시 ‘단계명\_\_파라미터명’ 형식으로 하이퍼파라미터 탐색 가능

**• 단계1: StandardScaler() (Linear/Ridge/Lasso 모델)**– 역할: 각 수치형 특성의 평균을 0, 표준편차를 1로 변환  
– fit(X\_train) → train 세트의 평균·분산 학습  
– transform(X\_train, X\_test) → train 기준으로 train/test 모두 스케일링

**• 단계2: 회귀 모델**  
– LinearRegression(fit\_intercept=True/False, positive=False/True)  
– Ridge(alpha=1.0, solver='auto')  
– Lasso(alpha=0.1, max\_iter=1000)  
– 각 모델의 주요 파라미터는 GridSearchCV로 튜닝 가능

• DecisionTreeRegressor 모델  
– 트리 기반 모델은 입력 스케일에 무관하므로 스케일러 단계 생략  
– Pipeline 단일 단계로 ('model', DecisionTreeRegressor(max\_depth=5)) 만 정의

학습 코드 요약

# 특성·타겟 설정

X ← clean[features]; y ← clean['CGPA\_numeric']

# 데이터 분할

X\_train, X\_test, y\_train, y\_test ← train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 파이프라인 정의

pipelines ← {

'LR' : Pipeline([('scaler', StandardScaler()), ('model', LinearRegression())]),

'Ridge': Pipeline([('scaler', StandardScaler()), ('model', Ridge())]),

'Lasso': Pipeline([('scaler', StandardScaler()), ('model', Lasso())]),

'Tree': Pipeline([('model', DecisionTreeRegressor(max\_depth=5))])

}

# 모델 학습

for name, pipe in pipelines:

pipe.fit(X\_train, y\_train)

# 평가 (RMSE, MAE, R²)

for name, pipe in pipelines:

y\_pred ← pipe.predict(X\_test)

calculate RMSE, MAE, R2

📌 **4. 성능 평가**

**4-1. 사용한 지표**

* **RMSE** (Root Mean Squared Error): 예측 오차의 제곱 평균의 제곱근. 값이 작을수록 모델 예측이 실제값에 근접함.
* **MAE** (Mean Absolute Error): 예측 오차 절대값의 평균. 직관적으로 오차 크기를 파악하기 용이.
* **R²** (결정계수): 1에 가까울수록 모델 설명력이 높음. 음수일 경우 단순 평균 예측보다 못함을 의미.

라인, 그래프, 도표, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**라인, 그래프, 도표, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**라인, 그래프, 스크린샷, 도표이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**라인, 그래프, 스크린샷, 도표이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**4-2. 예측 결과 시각화**

* **Actual vs Predicted 산점도**
  + 붉은 점선(y = x)은 완벽 예측을 나타냄.
  + 점들이 이 선에 가까울수록 예측 정확도가 높음.
* **Residual Plot (잔차 플롯)**
  + 가로축은 예측값, 세로축은 실제값–예측값(잔차).
  + 잔차가 0을 중심으로 고르게 분포하면 편향 없는 예측을 뜻함.

**4-3. 모델별 평가 요약**

|  |  |  |  |
| --- | --- | --- | --- |
| **모델** | **RMSE** | **MAE** | **R²** |
| LinearRegression | 1.317 | 0.994 | –0.803 |
| Ridge | 1.185 | 0.786 | –0.461 |
| Lasso | 1.048 | 0.595 | –0.142 |
| DecisionTree | 1.085 | 0.667 | –0.225 |

**4-4. 해석**

1. **전반적 비교**
   * 네 모델 모두 R²가 음수로, 단순 평균 예측보다 설명력이 부족함.
   * **Lasso**가 RMSE·MAE 최저, R² 최고(–0.142)로 가장 안정적인 성능을 보임.
   * **LinearRegression**은 RMSE·MAE 최고, R² 최저(–0.803)로 가장 부진.
2. **과소·과대 예측 패턴**
   * 모든 모델에서 실제 CGPA가 높은 구간(양의 값)일수록 예측값이 평균으로 끌리는 경향(under-regression)이 발견됨.
   * Residual Plot에 예측값이 클 때 잔차가 음수(예측값이 실제값보다 큼), 작을 때 양수(예측값이 실제값보다 작음)로 치우쳐 있음.
3. **잔차 분포**
   * **Lasso**와 **Ridge**의 잔차는 0 근처에 비교적 집중되어 있어 예측 안정성이 높음.
   * **DecisionTree**는 몇 개의 분할 값에 잔차가 몰리는 편향이 보임.
   * **LinearRegression**은 잔차 분포가 넓고 편향이 심해 추가 규제나 비선형 모델 적용이 필요함.

📌 **5. 하이퍼파라미터 튜닝**

**5-1. 튜닝 방법**

* **도구**: GridSearchCV (5-폴드 교차검증, scoring='neg\_root\_mean\_squared\_error')
* **목표**: 테스트 세트 RMSE 최소화
* **대상 모델 및 탐색 공간**
  + **LinearRegression**
    - fit\_intercept: [True, False]
    - positive: [True, False]
  + **Ridge**
    - alpha: [0.1, 1.0, 10.0, 100.0]
    - solver: ['auto','svd','cholesky']
  + **Lasso**
    - alpha: [0.1, 1.0, 10.0]
    - max\_iter: [1000, 5000]
  + **DecisionTreeRegressor**
    - max\_depth: [None, 3, 5]
    - min\_samples\_leaf: [1,4]
    - criterion: ['squared\_error','friedman\_mse']
    - splitter: ['best','random']

**5-2. 최적 하이퍼파라미터 & CV 성능**

|  |  |  |
| --- | --- | --- |
| **모델** | **최적 파라미터** | **CV RMSE (best)** |
| **LinearRegression** | {'model\_\_fit\_intercept': False, 'model\_\_positive': False} | 1.336 |
| **Ridge** | {'model\_\_alpha': 100.0, 'model\_\_solver': 'auto'} | 0.885 |
| **Lasso** | {'model\_\_alpha': 1.0, 'model\_\_max\_iter': 1000} | 0.877 |
| **DecisionTree** | {'model\_\_criterion':'friedman\_mse','model\_\_max\_depth':3,'model\_\_min\_samples\_leaf':4,'model\_\_splitter':'random'} | 0.891 |

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**5-3. 튜닝 결과 분석 및 파라미터 해석**

* **LinearRegression**
  + fit\_intercept=False 로 절편 항을 제거하고, positive=False 로 계수 부호 제약을 해제
  + 모델 단순화 시도에도 불구, CV RMSE가 오히려 증가(1.317→1.336)하여 성능 개선 미흡
* **Ridge**
  + alpha=100 의 강한 L2 규제를 적용해 회귀 계수의 분산을 대폭 억제
  + CV RMSE 1.185→0.885 로 크게 개선, 과적합이 효과적으로 완화됨
* **Lasso**
  + alpha=1.0 의 L1 규제로 일부 특성 계수를 0으로 축소하여 불필요한 변수를 제거
  + tol=0.01, max\_iter=1000 으로 수렴 조건 완화·최대 반복수 조정
  + CV RMSE 1.048→0.877 로 가장 낮은 오류 달성, 변수 선택과 노이즈 억제가 조화로운 결과
* **DecisionTreeRegressor**
  + max\_depth=3 로 가지치기 제한, min\_samples\_leaf=2 로 작은 잎 노드 제거
  + criterion='friedman\_mse', splitter='random' 로 분할 다양성 확보
  + CV RMSE 1.085→0.892 로 일정 개선, 과도한 분할을 방지하며 일반화 성능 상승

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **모델** | **Before Tuning RMSE** | **After Tuning RMSE** | **MAE Before → After** | **R² Before → After** |
| LinearRegression | 1.317 | 1.331 | 0.994 → 1.013 | –0.803 → –0.841 |
| Ridge | 1.185 | 1.019 | 0.786 → 0.550 | –0.461 → –0.079 |
| Lasso | 1.048 | 1.003 | 0.595 → 0.571 | –0.142 → –0.046 |
| DecisionTree | 1.085 | 1.037 | 0.667 → 0.604 | –0.225 → –0.118 |

* **Ridge** 와 **Lasso** 는 튜닝 후 RMSE·MAE가 유의미하게 하락하고 R²가 크게 개선되어 일반화 성능이 향상됨
* **DecisionTree** 도 과도한 분할을 제어함으로써 소폭의 성능 향상을 보임
* **LinearRegression** 은 단순 구조의 한계를 넘어설 수 없어, 튜닝 전후 모두 성능 개선이 어려웠음
* 규제 기반의 **Ridge/Lasso** 모델이 과적합 방지 및 노이즈 억제 측면에서 가장 효과적이었다.

📌 **6. 결론 및 고찰**

**6-1. 최종 모델 성능 종합 평가**  
• **선형 회귀 계열**  
o LinearRegression 은 RMSE 1.331, MAE 1.013, R² –0.841 로 기준(평균값 예측)보다 성능이 크게 뒤떨어짐.  
o Ridge (α=100) 와 Lasso (α=1.0) 규제 모델은 RMSE를 각각 1.019, 1.003, MAE를 0.550, 0.571까지 낮추어, 적절한 과적합 억제가 모델 성능 개선에 기여함을 확인.  
o 그럼에도 R²가 여전히 음수(–0.079, –0.046)인 것은, 입력 변수만으로는 CGPA 변동의 상당 부분을 설명하기에 정보가 부족함을 의미.

• **비선형 결정트리**  
o DecisionTreeRegressor (max\_depth=3) 는 RMSE 1.037, MAE 0.604, R² –0.118 로 회귀 계열보다는 약간 열세.  
o 트리 기반 모델 특성상 일부 패턴은 포착하나, 전반적 일반화 능력은 제한적임.

→ **종합:** Lasso 모델이 RMSE·MAE 기준으로 가장 낮은 오류를 기록했으나, 모든 모델의 R²가 음수에 머문 것은 ‘Age·Study\_Year·정신건강 이진지표’ 등 현재 특성만으로는 CGPA를 충분히 예측하기 어렵다는 한계를 반영한다.

**6-2. 데이터 및 모델의 한계**  
**• 특성 정보의 부족**  
o 주요 입력변수 간 상관관계가 매우 낮고, CGPA와도 거의 무상관(ρ≈0.01) → 결정 인자로 작용할 만한 강력한 예측 변수가 결여  
**• 범주형 변수 과다 단순화**  
o 이진화된 정신건강 지표만으로는 복합적 학업 성과를 설명하기 어려움

**6-3. 실생활 응용 가능성 및 확장 방향**  
• **학생 성취도 예측 보조도구**  
o 예비 진단용으로 사용하되, 실제 활용 시 추가 설문(공부시간, 수업 참여도, 과제 제출율 등)을 결합

• **멘탈 헬스 모니터링**  
o 정신건강 설문과 학업 성과 간 상관을 지속 추적하여, 위기 학생 조기 개입 시스템 연계

**6-4. 다음 단계에서 고려할 점**  
• **특성 엔지니어링 강화**  
o 교호작용 항(Age×Study\_Year), 다중 설문 합산 점수, 온라인 학습 로그 등 파생 변수 추가

• **고도화된 앙상블 모델**  
o RandomForest, GradientBoosting, XGBoost 등 비선형 앙상블 기법 적용

• **이상치 및 결측 처리 고도화**  
o Isolation Forest, KNN Imputer 등으로 극단치·결측 보완

**📌 7. 데이터셋 출처**• Kaggle – Student Mental Health  
https://www.kaggle.com/datasets/shariful07/student-mental-health

**7-1. 공식 문서**

1. 하이퍼파라미터 튜닝 (GridSearchCV / RandomizedSearchCV)  
   <https://scikit-learn.org/stable/modules/grid_search.html>
2. Pipeline & ColumnTransformer  
   <https://scikit-learn.org/stable/modules/compose.html>
3. GridSearchCV API 레퍼런스  
   <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>
4. RandomizedSearchCV 설명  
   <https://scikit-learn.org/stable/modules/grid_search.html#randomized-parameter-search>

**7-2. 참고 블로그**

1. Fadekemi Akinduyile. Effects of Mental Health on Students’ CGPA. GitHub.  
   <https://github.com/FadekemiAkinduyile/Effect-of-Mental-Health-on-Students-CGPA>
2. Hasar W. Student Mental Health Analysis. Kaggle Notebook.  
   https://www.kaggle.com/code/hasarw/student-mental-health
3. Dsaint31’s Blog – ColumnTransformer 사용법  
   https://dsaint31.tistory.com/828 [Dsaint31's blog](https://dsaint31.tistory.com/828?utm_source=chatgpt.com)
4. 김마아데이터 – 회귀모델 평가지표 (MSE, MAE, RMSE, R²)  
   https://kimmaadata.tistory.com/32 [Data Scientist Kimmaa&#039;s log](https://kimmaadata.tistory.com/32?utm_source=chatgpt.com)
5. 혼공머신 (Hyeonql) – 머신러닝 회귀 알고리즘 완벽 가이드 (Linear, Ridge, Lasso 등)  
   https://hyeonql.tistory.com/entry/%ED%98%BC%EA%B3%B5%EB%A8%B8%EC%8B%A0-2%EC%A3%BC%EC%B0%A8-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-%ED%9A%8C%EA%B7%80-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-%EC%99%84%EB%B2%BD-%EA%B0%80%EC%9D%B4%EB%93%9C-K-%EC%B5%9C%EA%B7%BC%EC%A0%91-%EC%9D%B4%EC%9B%83%EB%B6%80%ED%84%B0-%EC%84%A0%ED%98%95-%ED%9A%8C%EA%B7%80-%EB%A6%BF%EC%A7%80%EC%99%80-%EB%9D%BC%EC%8F%98%EA%B9%8C%EC%A7%80

**7-3. 사용한 주요 라이브러리**

**실행 환경:** 연구실 리눅스 서버 (VS Code SSH 원격 접속)

**접속 방식:** VSCode Remote-SSH / 아나콘다 가상환경에서 실행

주요 라이브러리 버전

|  |  |
| --- | --- |
| **라이브러리** | **버전** |
| Python | 3.9.21 |
| conda | 24.11.3 |
| numpy | 2.0.2 |
| pandas | 2.2.3 |
| scikit-learn | 1.6.1 |
| matplotlib | 3.9.4 |
| seaborn | 0.13.2 |

**📌 8. 부록**  
• 주요 코드 스니펫

**1. 파이프라인 정의**

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression, Ridge, Lasso

from sklearn.tree import DecisionTreeRegressor

pipelines\_dict = {

'LinearRegression': Pipeline([

('scaler', StandardScaler()), # 수치형 변수 스케일링

('model', LinearRegression()) # 선형 회귀 모델

]),

'Ridge': Pipeline([

('scaler', StandardScaler()),

('model', Ridge(alpha=1.0)) # L2 규제

]),

'Lasso': Pipeline([

('scaler', StandardScaler()),

('model', Lasso(alpha=0.1)) # L1 규제

]),

'DecisionTree': Pipeline([

('model', DecisionTreeRegressor(max\_depth=5)) # 트리 기반 모델

])

}

**2. EDA 시각화**

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import gaussian\_kde

# 이상치 1~99 분위수로 필터링

q1, q3 = np.percentile(data['CGPA\_numeric'], [1, 99])

filtered = data.loc[

(data['CGPA\_numeric'] >= q1) & (data['CGPA\_numeric'] <= q3),

'CGPA\_numeric'

]

# CGPA 히스토그램 + KDE

plt.figure()

plt.hist(filtered, bins=20, density=True) # 히스토그램

x\_vals = np.linspace(filtered.min(), filtered.max(), 100)

kde = gaussian\_kde(filtered)

plt.plot(x\_vals, kde(x\_vals)) # KDE 곡선

plt.title('CGPA\_numeric 분포 (이상치 제거)')

plt.xlabel('CGPA')

plt.ylabel('밀도')

plt.show()

# 상관관계 히트맵

corr = data[['Age', 'Study\_Year', 'CGPA\_numeric']].corr()

plt.figure()

plt.imshow(corr, aspect='auto')

plt.colorbar()

plt.xticks(range(len(corr)), corr.columns, rotation=45)

plt.yticks(range(len(corr)), corr.index)

plt.title('수치형 변수 상관계수 히트맵')

plt.show()

**3. 성능 평가 루프**

metrics = {

'RMSE': lambda y, ŷ: np.sqrt(mean\_squared\_error(y, ŷ)),

'MAE': mean\_absolute\_error,

'R2': r2\_score

}

for name, model in pipelines.items():

y\_pred = model.predict(X\_test)

print(name, metrics['RMSE'](y\_test, y\_pred),

metrics['MAE'](y\_test, y\_pred),

metrics['R2'](y\_test, y\_pred))

**4. 하이퍼파라미터 튜닝**

metrics = {

'RMSE': lambda y, ŷ: np.sqrt(mean\_squared\_error(y, ŷ)),

'MAE': mean\_absolute\_error,

'R2': r2\_score

}

for name, model in pipelines.items():

y\_pred = model.predict(X\_test)

print(name, metrics['RMSE'](y\_test, y\_pred),

metrics['MAE'](y\_test, y\_pred),

metrics['R2'](y\_test, y\_pred))

from sklearn.model\_selection import GridSearchCV

param\_grids = { … } # 위 그리드 참조

best\_estimators = {}

for name, pipe in pipelines.items():

grid = GridSearchCV(pipe, param\_grids[name],

cv=5,

scoring='neg\_root\_mean\_squared\_error',

n\_jobs=-1)

grid.fit(X\_train, y\_train)

best\_estimators[name] = grid.best\_estimator\_