**□ 모듈프로젝트 산출물**

클라우드 AI 융합 전문가 양성 과정

|  |
| --- |
| 모듈 프로젝트 3  **- 온라인 머신러닝 대회 –**   * 1조 : 김원일, 신성진, 천성신, 최숙희 - |

**2021년 9월 30일**

**클라우드 AI 융합 전문가**

이름 : 최숙희

|  |
| --- |
| **목 차**  Ⅰ. 참가 대회  1.1. 대회 선정 이유  1.2. 컬럼 설명  Ⅱ. 데이터 전처리  2.1. 특성별 기술통계 및 시각화  2.2. 데이터 정규화  2.3. 컬럼 추가  2.3.1. ‘본사대면근로자수’  2.3.2. ‘밥’, ‘국’, ‘주요반찬’  2.4. 알고리즘에 적용한 컬럼  2.5. 다중공선성 처리  Ⅲ. 알고리즘 적용  3.1. Decision Tree Regression  3.2. Random Forest Regression  3.3. XGBoost Regression  3.4. Light GBM Regression  3.5. KNN Regression  3.6. Logistic Regression  3.7. Multiple Linear Regression  3.8. Polynomial Regression  3.9. Ridge Regression  3.10. Lasso Regression  3.11. Elastic Net Regression  Ⅳ. 대회 제출  Ⅴ. 소감문 |

**Ⅰ. 참가 대회**

* 1. **대회 선정 이유**
* **정형 데이터 기반**
* **딥러닝을 제외한 머신러닝 방식을 적용해서 해결가능**
* **시계열에 따른 분석 불필요**

|  |
| --- |
|  |

* 1. **컬럼 설명**

|  |
| --- |
| 컬럼은 날짜를 알 수 있는 ‘일자’, ‘요일’과, 사옥 내 인원수에 관련된 ‘본사정원수’, ‘본사휴가자수’, ‘본사출장자수’, ‘본사시간외근무명령서승인건수’, ‘현본사소속재택근무자수’, 그리고 메뉴에 관한 부분인 ‘조식메뉴’, ‘중식메뉴’, ‘석식메뉴’를 독립변수로 활용할 수 있고, 예측의 목표가 될 컬럼은 중식과 석식을 먹은 인원수인 ‘중식계’와 ‘석식계’입니다. |

**Ⅱ. 데이터 전처리**

**2.1. 특성별 기술통계 및 시각화**

|  |
| --- |
| 특성들의 기술통계와 시각화를 살펴보았을 때, 두드러지는 특징은 없었지만, ‘일자’컬럼을 연,월,일로 나누어서 보았더니, ‘연도’의 경우 ‘석식계’값이 하락하는 추세를 보이는 등의 특성을 띄고 있어서 이런 방식으로 나누어서 활용하기로 결정하였습니다.    결측치를 지닌 특성은 없었습니다. 그러나, ‘현본사소속재택근무자수’ 컬럼의 기술통계치를 살펴봤을 때, 대부분의 값이 0임에도 불구하고, 최대값이 533으로 편차가 매우 크다는 점을 확인했습니다.  ‘재택근무자’가 발생했을 때의 데이터만 조회해봤더니, 코로나19가 심각해진 2020년 초부터 여서 데이터 입력에 문제가 있었다고 보이지는 않았습니다. 따라서 별도의 데이터 조작은 하지 않았습니다. |

**2.2. 데이터 정규화**

|  |
| --- |
| 오른쪽 그래프는 목표변수를 로그 취한 것이고, 왼쪽은 원본 데이터의 분포입니다. 저희가 사용한 데이터의 경우, Train data의 목표변수가 원래도 정규분포에 가까운 모양새였지만 값에 로그를 취해주면 모델의 예측 성능이 조금 더 올라가는 것을 확인해서 로그함수를 이용해서 데이터를 정규화 하였습니다. |

**2.3. 컬럼 추가**

**2.3.1. ‘본사대면근로자수’**

|  |
| --- |
| 기존에 원래 존재하는 ‘본사정원수’에서 ‘본사휴가자수’, ‘본사출장자수’, ‘현본사소속재택근무자수’를 빼주어서 일자별로 사옥에 실제로 출근한 사원의 수를 나타내 줄 ‘본사대면근로자수’ 컬럼을 만들었습니다. |

**2.3.2. ‘밥’, ‘국’, ‘주요반찬’**

|  |
| --- |
| ‘중식메뉴’와 ‘석식메뉴’ 컬럼을 이용해서 3개의 컬럼을 추가했습니다. 메뉴가 들어있는 컬럼을 살펴보면, 밥류 – 국류 – 반찬들이 나타나 있고, 반찬은 주요하다고 할 수 있는 것들부터 기본 반찬인 김치 등의 순서로 나열되어 있습니다. 이를 밥, 국, 그리고 반찬들 중에 가장 앞부분에 나오는 값을 ‘주요반찬’ 이라는 컬럼의 값으로 할당했습니다.    카레라이스 등 특색있는 밥류가 아닌 쌀밥/현미밥 등 일반적인 밥종류를 나열한 값들은 모두 ‘쌀밥＇으로 대체했습니다.    새롭게 만든 메뉴와 관련된 컬럼들은 회귀분석에 이용할 수 있도록 정수로 표현된 범주형으로 바꿔주었습니다. |

**2.4. 알고리즘에 적용한 컬럼**

|  |
| --- |
| Random Forest를 이용해서 컬럼의 중요도를 분석해 본 후에 중식계 예측을 위한 알고리즘 적용에 ‘밥’컬럼은 사용하지 않았습니다.      Train data에서 목표변수를 제외하고, 새로운 컬럼으로 나누어준 컬럼도 제외하고, ‘조식메뉴’와 같이 예측에 영향이 없을 것으로 보이는 컬럼도 제외한 채로 알고리즘에 적용했습니다. |

**2.5. 다중공선성 처리**

|  |
| --- |
| MinMaxScaler()를 이용해서 데이터의 범위를 통일해준 다음, variance\_inflation\_factor()를 이용해서 컬럼별 다중공선성을 확인하고 10을 초과하는 컬럼 중에서 값이 가장 큰 컬럼을 삭제하는 작업을 반복했습니다.      다중공선성이 큰 컬럼들을 삭제하고 남은 컬럼은 위와 같습니다. 그러나 다중공선성을 처리한 컬럼을 사용한 경우보다 다중공선성이 크게 나타난 컬럼일지라도 활용하는 편이 회귀모델들의 예측성능이 높았습니다. |

**Ⅲ. 알고리즘 적용**

**3.1. Decision Tree Regression**

|  |  |  |  |
| --- | --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - 독립변수가 되는 각 컬럼의 값에 규칙을 적용하면서 독립 변수 공간을 분할하는 모형  - 전체 학습 데이터 집합(부모 노드)을 각 독립 변수의 값이 기준 값보다 작은 데이터 그룹(자식 노드 1)과 해당 독립 변수의 값이 기준 값 이상인 데이터 그룹(자식 노드 2)으로 나누는 과정을 자식 노드에 한가지 클래스의 데이터만 존재할 때까지 반복 |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | Decision Tree를 사용할 때는, 트리의 깊이가 일정수준 이상으로 깊어지면 과적합이 발생하여,  오히려 모델의 예측성능이 떨어지므로 ‘Max\_depth’라는 파라미터를 조절하여 최적의 성능을 낼, 트리의 깊이를 찾는 것에 주의했습니다. |        |  | | --- | | 모든 모델의 성능을, 정확도는 R2(결정계수)로 확인하고, 오차는 MSE로 확인했습니다. | |

**3.2. Random Forest Regression**

|  |  |
| --- | --- |
| **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**       |  | | --- | | 수업시간에 배운 내용을 적용해서 Random Forest를 적용해봤습니다. | |

**3.3. XGBoost Regression**

|  |  |
| --- | --- |
| **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**           |  | | --- | | XGBoost는 수업시간에 배웠듯이 GridSearchCV()를 이용해서 최적의 파라메터를 찾을 수 있으므로, n\_estimators, max\_depth, random\_state, min\_child\_weight, colsample\_bytree, learning\_rate를 조절해서 모델의 예측성능을 높였습니다. | |

**3.4. Light GBM Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - Gradient Boosting 프레임워크로 Tree 기반 학습 알고리즘  - Light GBM은 **Tree가 수직적으로 확장**되는 leaf-wise  - 다른 Tree기반의 알고리즘은 Tree가 수평적으로 확장되는 level-wise  - 동일한 leaf를 확장할 때, leaf-wise 알고리즘은 level-wise 알고리즘보다 더 손실을 줄일 수 있다. |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | Light GBM은 100개가 넘는 파라메터를 적용할 수 있어서 파라메터 튜닝이 까다로운 알고리즘 이었고, 저희조는 기본이 되는 파라미터들을 중심으로 조절했습니다.  사용한 파라미터는, Tree의 잎사귀의 수로를 의미하며 Tree의 복잡성을 컨트롤하는 ‘num\_leaves’, 하나의 잎사귀가 가질 데이터의 최소치를 의미하고 과적합을 막기위해 사용하는 ‘Min\_data’, 범주형 특성의 인덱스를 받는 ‘Categorical feature’, 나무를 반복하며 부스팅할 횟수인 ‘Num\_iterations’, 부스팅 방법을 고르는 ‘boosting’을 조절했습니다.  Boosting은 기본값일 때 가장 성능이 좋았고, Num\_iterations는 기본값인 100보다는 1,000이상으로 값을 높였을 때 성능이 가장 좋았습니다. | |

**3.5. KNN Regression**

|  |  |  |  |
| --- | --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - 데이터 셋이 **수백 개 이상의 특성**을 가졌거나 혹은 **특성 값에 0이 많은 경우**에는 적합하지 않음  - 예측이 느리므로 실무에서는 잘 사용되지 않음  - 이웃을 하나만 사용할 때는 훈련 세트의 각 데이터 포인트가 모든 선을 통과하게 되는데,  이는 모델이 지나치게 복잡함을 의미하며, 예측 성과가 좋지 못함.  반대로, **많은 수의 이웃**을 사용할 수록 훈련 데이터에는 잘 안 맞을 수 있지만 **높은 정확도**를 얻게 됨. |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | 데이터를 train과 test 데이터로 나눈 후, KNN 트리를 생성 후 데이터들의 각각의 정확도를 테스트 합니다. |      |  | | --- | | For 문을 사용하여, 이웃 수에 따른 각각의 정확도와 MSE의 값을 소수 5째 자리까지 출력했습니다. MSE가 가장 낮은 값에 속할 때는 이웃이 8일 때 0.02723으로 가장 낮았습니다. | |

**3.6. Logistic Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - 회귀를 사용하여 데이터가 어떤 **범주에 속할 확률을 0에서 1 사이의 값으로 예측**하고 그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 지도 학습 알고리즘  - **시그모이드함수(sigmoid function)**를 이용해서 특성 값을 음의 무한대부터 양의 무한대까지의 실수 값을 **0부터 1사이의 실수** 값으로 변형  - 여타 회귀분석 알고리즘과는 다르게 **target data가 범주형** 일 때 사용가능 |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | Logistic Regression회귀는 사용해봤으나, 실질적으로는 0과 1의 사이를 분류하는 알고리즘으로, 모델의 정확도만 체크 하였습니다. | |

**3.7. Multiple Linear Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - 독립변수 x와 종속변수 y간의 관계를 정량적으로 찾아내는 작업을 할 때, **독립 변수 x와 이에 대응하는 종속 변수 y간의 관계가 선형 함수** f(x)인 회귀 모델  - **수치형 독립변수들**과 **연속형 종속변수** 간의 관계를 선형으로 가정하고 이를 가장 잘 표현할 수 있는 **회귀계수를 추정** |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | Linear Regression의 파라메터는 모델의 상수항 여부를 결정하는 ‘fit\_intercept’와 훈련데이터가 원본에 다시 전달되는 것을 방지하는 ‘copy\_X’등이 있고, 이들을 조절해봤으나 모델 성능에 차이는 없었습니다. | |

**3.8. Polynomial Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - **다차원 모델식**을 이용한 회귀 분석 |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**       |  | | --- | | 다항회귀의 경우 이를 직접 수행해주는 라이브러리는 없고, ‘Polynomial Features’라는 객체를 만들어서 회귀계수의 차수를 지정하고, 그에 맞게끔 특성값을 변환한 후에 변형된 특성 값을 ‘단순 선형회귀’하는 방식으로 적용합니다.  단순 선형회귀와 2차원, 3차원의 다항회귀를 적용해본 결과, 2차원 다항회귀, 단순 선형회귀, 3차원 다항회귀 순으로 예측성능이 높았습니다. 그 이유는, 예측변수의 분포형태가 완전한 선형이 아니고 살짝 굽어져 있기 때문이라고 생각됩니다.  참고로, 데이터 정규화 전후로, 목표변수의 분포형태가 크게 바뀌지 않아서 그런지 데이터 정규화 유무에 따른, 성능차이는 크지 않았습니다. 그렇지만, 정규화를 했을 때 데이터 분포가 더 굽어진 점에서 알 수 있듯이, 다항회귀의 성능은 원본 데이터를 그대로 사용했을 때보다 정규화를 이용했을 때 더 높았습니다. | |

**3.9. Ridge Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - 선형 모델의 **과적합 방지**  - 과적합 발생시, 극단적으로 오르내리는 그래프가 생성이 되고, 이를 표현하는 **선형 회귀의 계수 값이 매우 큼**.  - 이런 Variance가 큰 상황을 막기 위해, 계수의 크기에 대해 **패널티를 주는 수식**을 추가 한 모델 |   **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | 최적의 alpha값을 찾기 위해서, RidgeCV() 객체를 이용해서, 0부터 99.99까지 0.01간격으로,  alpha값의 변화에 따른 모델성능을 점검한 후에 예측을 수행했습니다. | |

**3.10. Lasso Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | ‘계수의 절댓값’과 ‘alpha’를 곱한값을, ‘오차함수＇에 더해줍니다.  이렇게 되면, 1미만의 계수는 0이 되면서 해당 계수를 갖는 특성은 제거되게 됩니다.  이런 면에서 볼 때, Ridge는 전반적으로 모든 계수의 크기를 감소시키는 방식이고, Lasso는 불필요한 계수의 값을 0으로 만들어서 ‘적절한’ 특성만 회귀에 포함시키는 방식으로 작동한다는 차이가 있습니다. |     **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | Ridge회귀 보다 Lasso회귀가 미세하게 높은 정확도를 보였지만 큰 차이는 없었습니다. | |

**3.11. Elastic Net Regression**

|  |  |  |
| --- | --- | --- |
| **> 알고리즘 설명**   |  | | --- | | - 가중치의 절대값의 합(L1, Lasso)과 제곱 합(L2, Ridge)을 **동시에 제약 조건으로** 가지는 모형  - **분산과** **변수 모두 줄이기** 위해 사용  - Lasso 단점 보완 |     **> 알고리즘 실행한 코드 주요부분 캡쳐+설명**     |  | | --- | | Ridge와 마찬가지로,ElasticCV() 객체를 이용해서,최적의 alpha값을 찾은 후에,예측을 수행했습니다. | |

**Ⅳ. 대회 제출**

|  |
| --- |
| 대회제출을 위한 test데이터를 이용할 때, train data에 적용했던 것과 동일한 전처리과정을 밟았고, 예측값이 로그로 나오기 때문에 지수함수를 이용했습니다. 데이터를 정규화할 때, Log0은 음의 무한대이므로 파이썬 오류를 방지하기 위해서 Log를 취한 후에 1을 더해 주었기 때문에, 모델의 예측값을 지수함수에 대입한 후에 1을 뺀 값을 제출했습니다.    Test data를 이용한 예측을 데이콘에 제출한 결과가 가장 좋은 것은 Decision Tree를 이용한 예측이었습니다. |

**Ⅴ. 소감문**

|  |  |
| --- | --- |
| 성명 | 프로젝트 후 소감 |
| 김원일 | 3번째 프로젝트를 거쳐오면서 새로운 팀원들과 여러가지 프로젝트를 함께 해오며 팀장이라는 직책에 크게 신경 쓴 바가 없었습니다.  하지만 이번 프로젝트를 진행해 오면서 팀장의 역할이 크게 와 닿았습니다.  그 이유로는 여러 프로젝트에서 겪었던 바 팀원들간의 프로젝트 진행에 있어 비대면 프로젝트가 프로젝트 진행에 영향을 끼쳐 더뎌지는 상황이 종종 발생하곤 했습니다.  하지만 이번 프로젝트의 경우에는 팀장의 역량으로 팀원들에게 알맞은 일의 배분이 이루어졌었고 프로세스의 세분화로 정확한 일정의 소화가 가능했습니다.  부족한 부분에 대해서는 팀장이 팀원들을 도와 지정 날짜에 결과물을 도출해 나갈 수 있었고 결과물의 검토 또한 팀장이 도맡아 결과물 간의 연관성을 해치는 일 없이 프로젝트를 완성할 수 있었습니다.  프로젝트에서 팀장의 역할과 그 중요성에 대해서 알 수 있는 계기가 되었고 직책의 중요성에 대하여 생각할 수 있었습니다. |
| 신성진 | 벌써 세번째 프로젝트가 종료되었습니다.  처음에는 말을 나누는것조차 어색했던 첫 프로젝트 때와는 달리 세번째로 도달한 팀 프로젝트 시작은 무척이나 좋은 분위기의 출발 선에서 시작되었다고 생각됩니다.  주도적인 팀장의 지시 아래에서 각자 맡게된 다양한 알고리즘을 열정적으로 찾아보고, 매일 수차례씩 각자 알게 된 알고리즘 이해에 대한 부분을 공유하고, 어떤 방향으로 나아갈지 논의하며 다양한 알고리즘들을 공부 할 수 있었습니다.  이번 프로젝트의 목표인 머신러닝 대회를 참여하게 되면서 머신러닝에서는 어떠한 알고리즘을 사용하게 되는지, 피라메터를 얼마만큼 잘 활용하는지, 데이터의 전처리 작업이 얼마나 중요한지 확인 할 수 있었습니다.  이 세가지에 따라 같은 데이터임에도 끌어낼 수 있는 모델의 예측성능이 각각 다르거니와, 결과적으로 다른 값을 도출해 낼 수 있다는 알수 있었기 때문입니다.  이번 프로젝트에서는 프로젝트를 진행하며 교육 내용에 나와 있는 부분뿐만이 아니라, 좀 더 다양한 방법을 통해서 추가적 부분을 가미해서 공부를 해야한다는 필요성을 느낄 수 있었고, 좀 더 적극적으로 공부해야지만 스스로를 발전 시킬 수 있다는 것을 다시 한번 느낄 수 있었습니다.  또한 첫번째와 두번째 프로젝트에서 팀장직을 맡았지만 팀장이 아닌, 팀원으로서 지금의 팀장을 보게 되니 좀 더 지금의 팀장님처럼 주도적으로 팀원분들을 이끌어냈다면 이전의 프로젝트에서는 어땠을까? 라는 느낌이 이전의 내게 아쉬움으로 남습니다.  마지막으로 좋은 출발선에서 시작하여 마무리가 되었지만, 아무래도 팀원분들에게는 죄송스러움이 남습니다. 다른 팀원들에게 이번 프로젝트에 있어 더 좋은 결과물을 산출해 낼 수 있는 가능성을 개인적 사정으로 인해 저버린게 아닐까 싶다는 생각이 들기도 합니다.  앞으로 남은 네번째 프로젝트와 최종 프로젝트에서는 앞선 프로젝트들의 부족한 부분들을 바탕으로 미숙했던 부분을 매꿔가며 팀원과의 협업을 통해 열정적으로 참여하겠다는 마음가짐을 가질 수 있었습니다. |
| 천성신 | 벌써 3차 프로젝트를 무사히 마쳤습니다 이젠 새로운 팀들과 시작하는 방식이 점점 익숙해지면서 어떻게 바로 시작해야 하는지 모두가 알고 있다 보니 점점 프로젝트의 시작이 수월해지고 있는 것 같습니다 이번 프로젝트는 비대면으로 진행하였습니다 처음으로 해보는 비 대면 진행에 어려움이 많이 있지 않을까 하는 걱정을 했지만 팀장님의 지시와 팀원들이 적극적으로 참여한다면 비대면에도 불구하고 어렵지 않게 프로젝트를 진행할 수 있다는 것을 알게 되었습니다 코딩하다 어려움이 있더라도 공유화면과 질의응답이 빨라 비 대면 이였지만 모르는 부분 없이 공부하고 즐거웠습니다 남은 프로젝트도 이번과 같이 비 대면 진행을 하더라도 걱정하지 않고 참여할 수 있을 것 같습니다 |
| 최숙희 | 교육과정에 참여한 이후로 세 번째로 진행하는 모듈 프로젝트인 만큼 조를 이뤄서 하는 분업과 협업에 익숙해졌습니다. 비대면으로 프로젝트가 진행되었지만 하루에 4번씩 진행상황을 공유했기 때문에, 계획된 일정에 맞춰서 맡은 일을 진행하면서도 프로젝트의 앞으로의 방향을 위해 할 수 있는 일을 틈틈이 고민해볼 수 있었습니다.  이번 프로젝트를 준비하면서 온라인 머신러닝 대회를 처음으로 살펴보게 되었습니다. 머신러닝을 위해서는 알고리즘에 대한 이해가 선행되어야 하고, 알고리즘이 커버하고 있는 다양한 파라메터를 얼마나 잘 활용하는지가 중요하다고 생각하게 되었습니다. 이번 프로젝트를 진행하면서, 최대한 다양한 회귀 알고리즘에 대해 공부해보고자 했습니다.  그렇지만, 머신러닝에 있어서 가장 중요한 것은 데이터 전처리라고 생각하게 되었습니다. 왜냐하면 데이터 전처리 과정에서 ‘데이터의 이상치를 제거하는지 여부’, ‘로그등을 이용한 목표변수의 정규화 여부’, ‘특성 값들의 범위를 조정하는지 여부’, ‘각각의 특성과 목표변수의 상관관계를 살펴보고 알고리즘에 적용할 컬럼으로 무엇을 선택하는지’, ‘특성의 중요도를 파악할 수 있는 알고리즘을 1차적으로 적용해서, 예측에 이용할 특성을 재선별하는 과정을 거쳤는지 여부’ 그리고 ‘데이터를 정규화해서 예측한 경우, 예측값을 다시 조작해야 하는점’등 다양한 방식을 고려할 수 있고 이에 따라서 예측의 결과가 천차만별로 바뀌는 것을 확인했기 때문입니다. 또한, 목표변수와 상관계수가 0.2~0.1미만인 특성들은 제거하는 편이 예측성능을 높일 수 있다는 블로그 글들을 보았지만, 저희 조가 사용한 데이터의 경우 상관계수가 0.1대인 특성도 사용하는 편이 모델의 예측성능을 높였습니다. 이런 점들로부터, 책이나 블로그를 보면서 데이터 전처리의 일반적인 과정을 익힐 수 있더라도 실제로 데이터를 전처리 할 때는 일일이 다양한 시도를 해보며 모델의 예측성능을 확인해야 한다는 것을 알게 되었습니다. 이렇듯 데이터 전처리는 정해진 정답이 없기 때문에 머신러닝에서 가장 신경써야 할 단계라고 생각하게 됐습니다.  능동적이고 열정적인 조원들과 함께하게 되어서 프로젝트 내내 지치지 않을 수 있었습니다. 앞으로 남은 프로젝트에서 더욱 능동적으로 프로젝트에 참여하겠다는 다짐을 하게 되었습니다. |