Wrap-Up Report

1. 프로젝트 개요

1) 프로젝트 주제 및 목적

본 프로젝트는 AI와 머신 러닝을 활용해 **아파트 전세 실거래가를 예측**하는 프로젝트입니다.

한국에서 아파트는 주거 문화의 중심이자 주요 자산 증식 수단으로 오래전부터 자리 잡아왔습니다. 나아가, 전세 시장은 매매 시장과 밀접하게 연관되어 있어, 부동산 정책과 시장 예측의 중요한 지표가 됩니다.

이번 대회를 통해 **미래 시점의 부동산 전세 실거래가를 직접 예측함**과 동시에 **부동산 시장의 정보 비대칭성 해소를 기여**하고자 합니다.

2) 프로젝트 환경

- **서버 환경** : 4 GPU V100, VScode와 ssh key로 연결하여 활용
- **협업 tool** : Notion, GitHub, Slack, 카카오톡
- **기술 stack** : Python, Pandas, Scikit-Learn, Pytorch

3) 데이터 구조

- **데이터 양식**

`train.csv` : 약 180만건의 모델 훈련에 사용되는 전세 실거래가 훈련용 데이터(2019~2023)

`test.csv` : 약 15만건의 모델 성능 평가 및 실제 예측에 사용되는 전세 실거래가 평가용 데이터(2024)

`sample_submission.csv` : test.csv의 예측 결과 제출을 위한 샘플 양식

`interestRate.csv, subwayInfo.csv, schoolInfo.csv, parkInfo.csv` : 기타 활용 데이터(금리, 지하철, 학교, 공원)

---

2. 프로젝트 팀 구성 및 역할

이름	공통 역할	개별 역할
박재욱	EDA, Feature Engineering	DL Modeling(MLP, Transformer), Seed Ensemble
서재은	EDA, Feature Engineering	DL Modeling(CNN + MLP)
임태우	EDA, Feature Engineering	ML Modeling(XGBoost) + Hyperparameter Tuning(Optuna)
최태순	EDA, Feature Engineering	ML Modeling(XGBoost) + Feature Selection(Feature Importances)
허진경	EDA, Feature Engineering	ML modeling(XGBoost) + Spatial Weight Matrix

---

3. 프로젝트 수행 절차 및 방법

1) 타임라인

2) 프로젝트 세부 수행 절차

1. EDA
    a. 컬럼별 데이터 분포 확인
    b. 컬럼별 데이터 이상치 확인
2. Feature Engineering
    a. 파생 지표 생성
3. Data Preprocessing
    a. 이상치 처리 및 Standard Scaler로 정규화
4. Feature Selection
    a. 기존 모델(XGBoost)에서 Feature Importance 측정
    b. 중요도에 따라 최종적으로 상위 20개의 feature 선택
5. Model Training
    a. DL Model Test : CNN + MLP, MLP, TabTransformer
    b. ML Model Test : LightGBM, XGBoost, Stacking, Voting, Seed Ensemble
6. Model Evaluation
    a. Valid MAE, Public MAE를 통한 모델 평가
    b. 최종 모델 선택

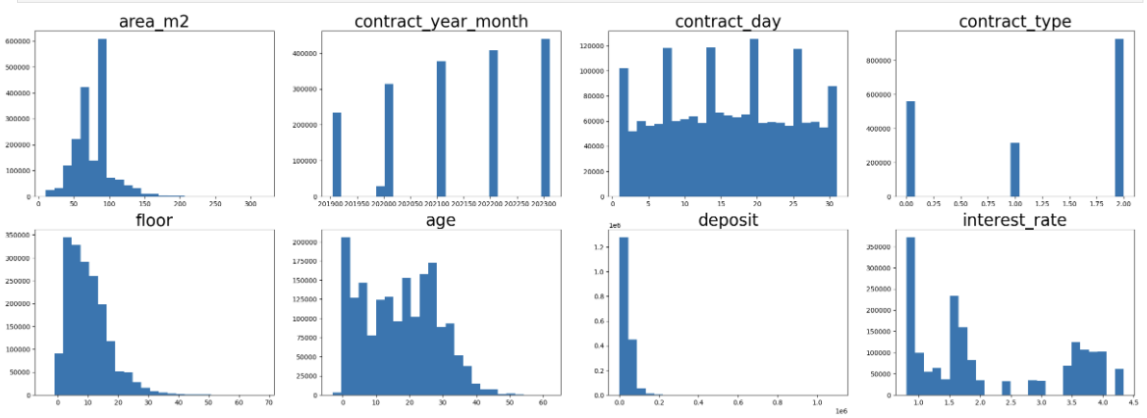
---

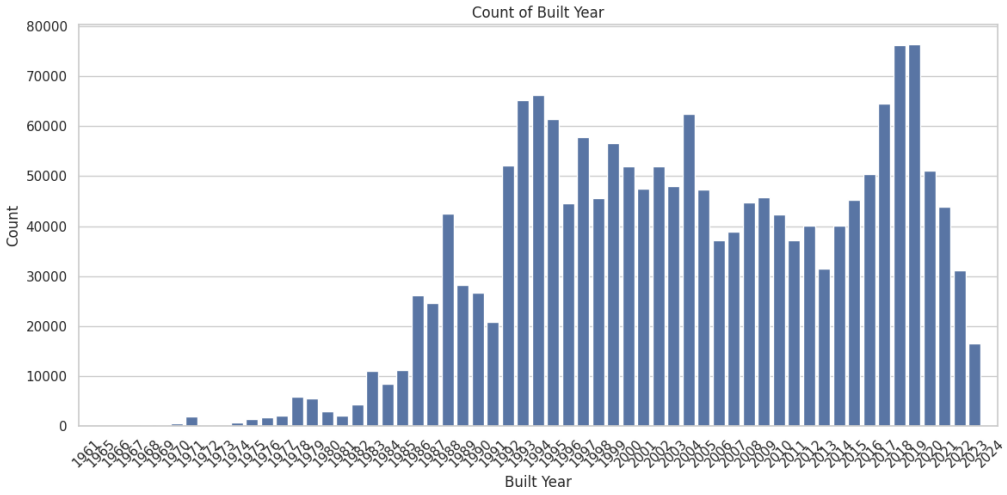
4. 프로젝트 수행 결과

1) 프로젝트 구조

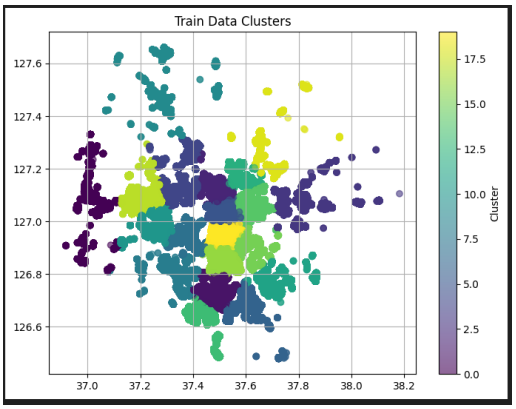
2) EDA 및 preprocessing

1. 컬럼 별 데이터 분포

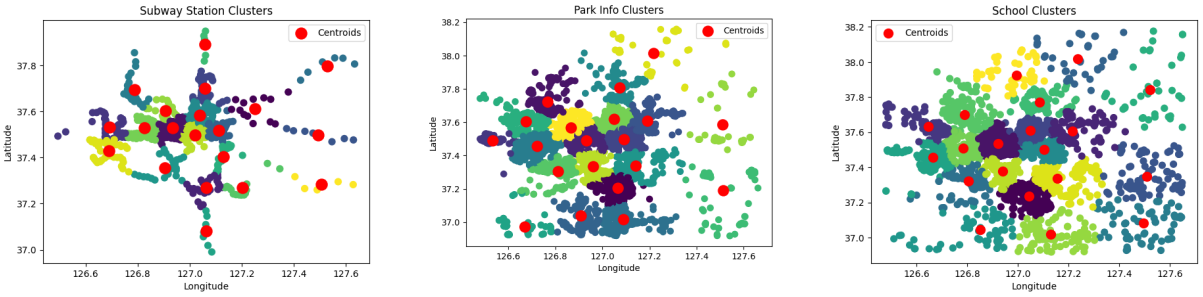




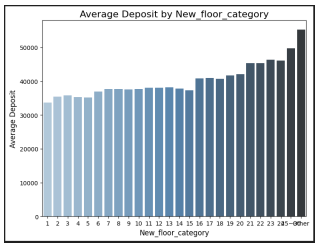
2. 서울 아파트 위치 정보 중심 클러스터링

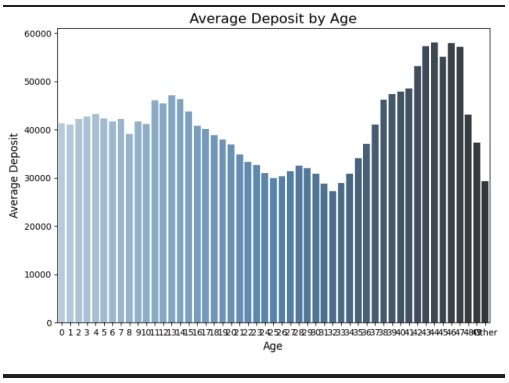


3. 공공시설 별(지하철, 공원, 학교) 클러스터링 (n=20)

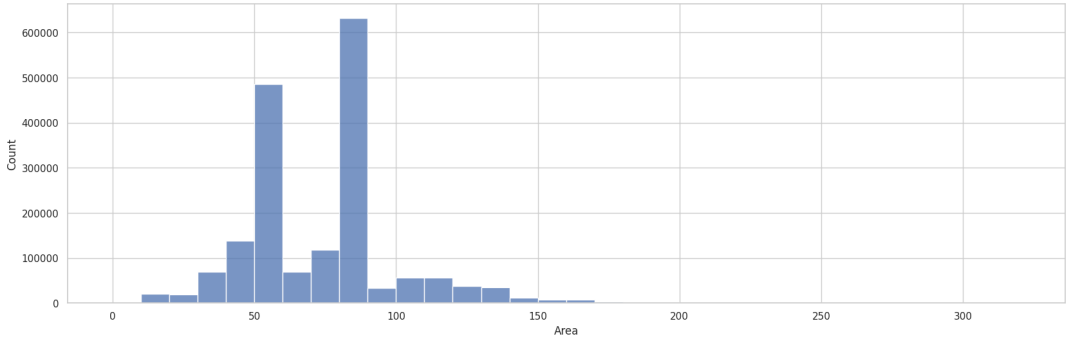


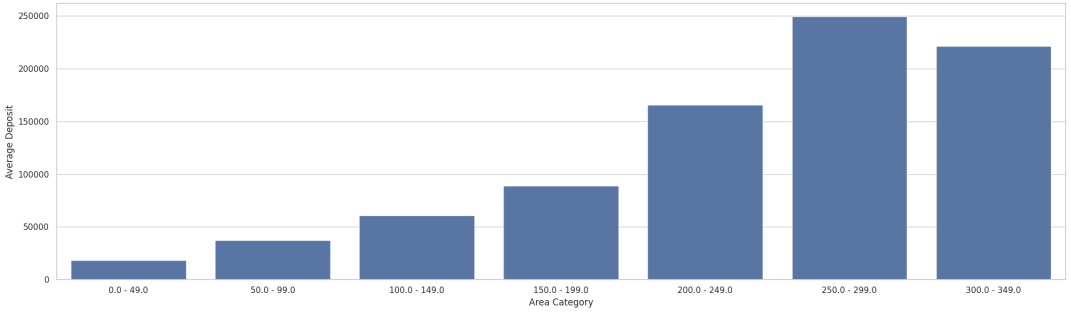
4. floor, age 변수의 범주화 진행(추가로 area도 진행함)



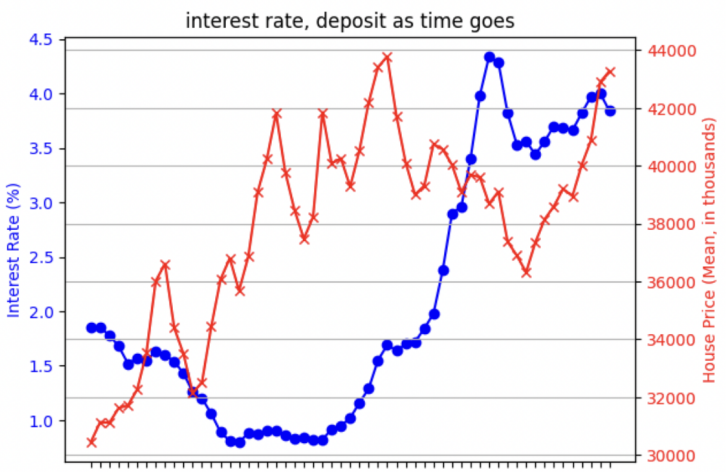


5. area_m2 범주화 및 범주별 평균 전세가





6. 금리와 전세가의 관계 파악



3) feature engineering

각자 EDA한 결과를 바탕으로 feature들을 추가 생성했습니다.

feature engineering 변수

Aa function	column name	description
clustering	subway_info, park_info, school_info	subway, park, school의 위경도 기준으로 k=20 clustering
create_clustering_target	cluster, distance_to_centroid, target_encoded_price_per_area, target_encoded_deposit	target을 기준으로 k=20 clustering, 해당 cluster 별 centroid와의 거리, 클러스터별 평균 면적당 전세가, 클러스터 타겟 인코딩
create_cluster_deposit_median	cluster_median	target 기준의 cluster 중앙 전세값
transaction_count_function	transaction_count_last_3_months	3 개월 동일한 아파트 거래량
create_subway_within_radius	subways_within_radius	반경 내 지하철 개수 (default = 0.01km)
create_school_within_radius	schools_within_radius	반경 내 학교 개수 (default = 0.02km)
create_school_counts_within_radius_by_school_level	elementary_schools_within_radius, middle_schools_within_radius, high_schools_within_radius	반경 이내 초,중,고 개수
create_place_within_radius	public_facility_count	반경 이내 공공시설 개수(default = 0.01km
create_nearest_subway_distance	nearest_subway_distance	가장 가까운 지하철까지의 거리
create_nearest_park_distance_and_area	nearest_park_distance, nearest_park_area	가장 가까운 공원까지의 거리, 면적
create_nearest_school_distance	nearest_elementary_distance, nearest_middle_distance, nearest_high_distance	가장 가까운 초,중,고까지 거리
weighted_subway_distance	weighted_subway_distance	환승역 가중치 거리 계산

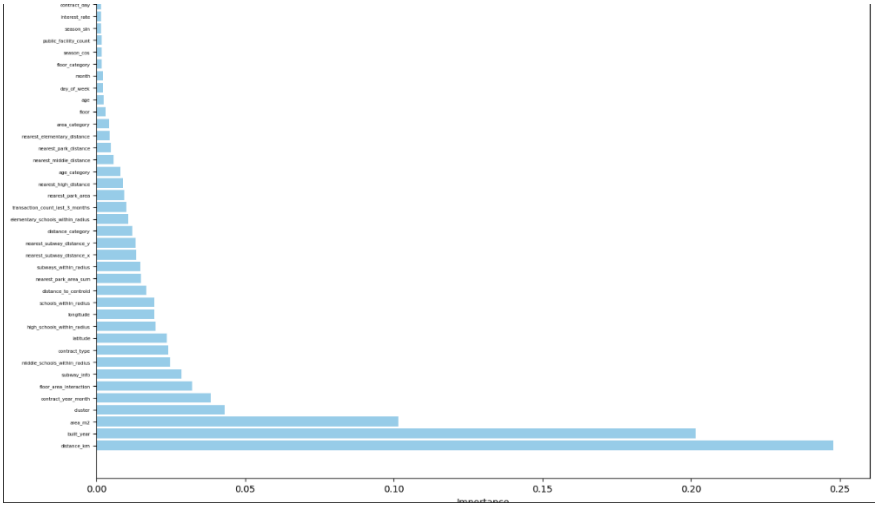
Aa function	column name	description
create_temporal_feature	year,month,date,quarter,season ...	year, month, date 조작 변수
create_sin_cos_season	season_sin & cos	계절 mapping 후 sine, cosine 적용
create_floor_area_interaction	floor_area_interaction	층수와 면적의 관계
distance_gangnam	distance_km, distance_category	강남까지의 거리 및 범주화
create_sum_park_area_within_radius	nearest_park_area_sum	반경 이내 공원 면적의 합
shift_interest_rate_function	interest_rate_year or month	계약 시점 기준 이전 금리(default = 3,6,12)
categorization	age_category, floor_category, area_category	age, floor, area범주
add_recent_rent_in_building	recent_rent_in_building	동일한 아파트(위도, 경도, 건축연도, 면적)의 최근 전세가
add_avg_rent_in_past_year	avg_rent_in_past_year	동일한 지역(위도, 경도, 건축연도, 면적)의 과거 평균 전세가(중앙값으로 계산)
add_rent_growth_rate	deposit_rate	연도별 최근 전세가 상승률

4) 최종 feature select

xgboost의 1차 feature importance로 top-20 feature를 최종 모델링에 사용할 변수로 선정했습니다.

- **사용 features(top-20)**

distance_km	floor_area_interaction	high_schools_within_radius	subways_within_radius
built_year	subway_info	longitude	nearest_subway_distance_x
area_m2	middle_schools_within_radius	schools_within_radius	nearest_subway_distance_y
cluster	contract_type	distance_to_centroid	distance_category
contract_year_month	latitude	nearest_park_area_sum	elementary_schools_within_radius



- **feature importance에 따라 상위 feature 10~25개를 선택했을 때의 결과 비교**

selected features count	MAE Scores
10	3840.6875
15	3719.1291
20	**3684.9866**
25	4483.9643

⇒ 결과를 비교해봤을 때 feature가 20일 때 가장 좋은 MAE 성능이 나와서 n을 20으로 선정했습니다.

5. Modeling

a. 학습 과정

- CNN + MLP 모듈

이 모델을 고안한 이유

- 기존 ML 모델에서 `latitude`, `longitude가` 중요한 피처임을 파악한 후 공간적 정보를 최대한 활용할 수 있는 모델을 만들고 싶었습니다.
- 우리가 서울시 지도를 2차원 평면으로 인식하듯 `latitude`와 `longitude`를 2차원 그리드로 재구성한 피처맵을 활용한다면 공간적 정보를 활용할 수 있을 것이라고 추측해, convolution 연산으로 공간적 정보를 파악하는 모델을 활용했습니다.

모델 구현 (`CombinedModel.py`)

- CNN, MLP model 단계에서 나온 output 두 가지를 concatenate 연결을 거쳐, fully connected layer를 통과해서 최종적으로 단일한 scalar값인 전세가를 예측합니다.

실험 결과

- MAE score가 낮아 최종 솔루션 모델로 채택하지는 못했습니다. 그러나, 실험을 거쳐 딥러닝 모델을 다룰 때 참고할 만한 인사이트를 얻을 수 있었습니다.

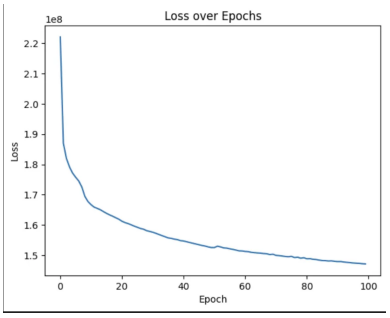
실험 1 (100epoch)

실험 2 (50epoch)

- MLP model의 linear 층을 2-> 1개로 축소
  - 모델의 복잡도를 줄이기 위한 조정

실험 3 (100epoch)

- 에폭 수를 100으로 늘려 진행
- valid MAE : **12712.6162**



- Loss 그래프가 안정적으로 떨어지지만, ML 모델을 사용했을 때 4000 정도이던 MAE score에 비교하면 한없이 큰 숫자였습니다.
- valid MAE : **14000**대

- 컬럼 개수 기존 50개 -> top 20으로 축소
- 카테고리 임베딩 차원 조절 128 -> min(50, int(col.nunique()+1)/2))
- valid MAE : **10877.6016**

- **XGBoost + Spatial Weight Matrix + Seed Ensemble**

XGBoost

- LightGBM, CatBoost 모델과 같은 트리 기반 모델 중에서도 가장 좋은 성능을 보여 채택하게 되었습니다.
- Feature Selection 과정에서 선택된 Feature를 바탕으로 학습을 진행하되, Optuna를 사용하여 최적의 하이퍼 파라미터를 설정했습니다.
- 단순히 가장 높은 점수를 보인 파라미터를 그대로 적용하기보다, 모델이 과적합되지 않도록 조정한 파라미터를 선택하여 일반화 성능을 향상시켰습니다.

Spatial Weight Matrix

- 데이터 샘플과 위도, 경도 거리 상 가까운 이웃의 종속 변수 값이, 해당 데이터 샘플의 종속 변수 값에 영향을 미칠 것이라는 가설을 바탕으로 공간적 가중치 행렬을 적용했습니다.
- Ball Tree 알고리즘을 이용하여 가까운 이웃 10개의 데이터 샘플을 구하고, 이웃과 해당 데이터 샘플 간의 거리를 측정하여 `1/거리` 를 가중치로 설정했습니다. 최종적으로 이렇게 구한 가중치를 10개의 가중치의 합으로 나누어 10개의 이웃에 대한 가중치의 합이 1이 되도록 했습니다.
- 이렇게 생성된 가중치 행렬과 전세가를 사용했을 때, 반영되지 않을 수 있는 면적과 전세가의 상관관계를 고려하기 위해 전세가를 면적으로 나눈 면적 당 전세가를 사용했습니다.
- 최종적으로 모든 데이터 샘플에 대한 면적 당 전세가와 가중치 행렬의 행렬곱을 통해 각 데이터 샘플의 가까운 이웃 10개에 대한 가중 평균을 피처로 추가했습니다.

Seed Ensemble

- Stacking, Voting Ensemble에 비해 Seed Ensemble이 보다 더 좋은 일반화 성능을 보여 최종 모델에 반영하게 되었습니다.
- 시드의 개수를 늘릴수록 일반화 성능이 향상되어 모델의 성능이 좋아지는 모습을 보였지만, 모델 학습 시간과 자원의 제약을 고려해 시드는 10개를 사용했습니다.
- 시드의 무작위성을 확보하기 위해 각각의 시드는 42, 7, 2023, 1024, 99, 512, 1001, 888, 3456, 1234를 사용했습니다.

- TabTransformer

이 모델을 고안한 이유

- 트리 모델에서는 dtype을 지정함에 따라 모델이 이를 인식하여 학습하지만 딥러닝 모델에서는 이를 인식하지 않고 float, int를 그 자체로 인식하기 때문에 categorical, numerical data를 효율적으로 해결할 수 있는 방법을 고민하였습니다.
- 여러 모델이 있었지만 TabTransformer는 이번 대회에서 주어진 테이블 데이터를 처리하는데 효과적이고 고민하고 있던 categorical, numerical 데이터를 처리하는데 효과적인 딥러닝 모델이라고 하여 채택하였습니다.

모델 구현 (`tabtransformer.py`)

- numerical data는 Min-Max scaler를 사용하여 정규화를 진행하고, categorical data는 임베딩 벡터화 시키고 numerical data와 마찬가지로 scale를 통일 시키기 위해서 추가로 Min-Max Saler를 진행하였습니다.
- 이후 두 데이터를 합친 다음 Fully Connected Layer를 거쳐 최종 output을 출력하게 됩니다.

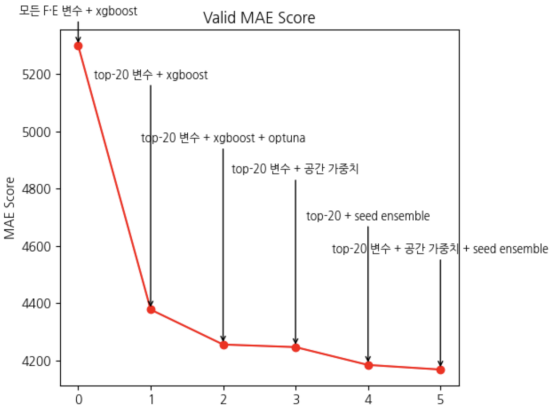
실험 결과

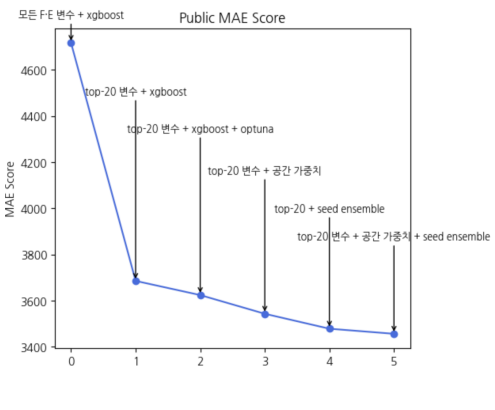
- 모델이 학습하는 과정에 있어서 과적합이 발생하여 `BatchNorm1d`, `dropout` 를 추가하여 과적합은 해결하여 train, valid loss가 전반적으로 하락하는 개형을 보여 제대로 학습하는 것 처럼 보였지만 평가 지표인 MAE를 봤을 때 이 역시 전반적으로 하락하는 형태를 가지지만 10,000 점 아래로 하락시키지 못하여 채택되지 못하였습니다.

b. 성능 기록

Method	Valid MAE	Public MAE
50epoch + CNN_MLP	10877.6016	9929.1350
100epoch + CNN_MLP	12712.6162	- (리더보드 마감으로 확인하지 못함)

Method	Valid MAE	Public MAE	Private MAE
모든 F·E 변수 + xgboost	5300	4716.4452	5555.9498
top-20 변수 + xgboost	4377.8782	3684.9866	4486.8749
top-20 변수 + xgboost + optuna	4255.8043	3623.6262	4410.5463
top-20 변수 + 공간 가중치	4246.8171	3542.5882	4327.4064
top-20 + seed ensemble	4184.5018	**3477.8167**	**4273.3032**
top-20 변수 + 공간 가중치 + seed ensemble	4168.3872	**3455.8046**	**4250.0153**





- **최종 제출 파일**

    - top-20 + seed ensemble(public MAE : 3477.8167)
    - top-20 + 공간 가중치 + seed ensemble(public MAE : 3455.8046)
- **선정 이유**
    - 최종 데이터셋에서 feature importance가 높은 변수들만 구성했을 때 좋은 성능을 보였기 때문에 해당 모델을 선정했습니다.
    - seed ensemble을 사용하여 일반화 성능이 높은 모델을 선정했습니다.
    - 공간 가중치 행렬로 계산된 새로운 feature가 추가되었을 때와 그렇지 않았을 때의 결과를 비교해보기 위해 두 파일을 선택했습니다.

6. 개선사항

- 전처리 모듈, 모델 클래스 종속성 개선
- 모델 학습 파이프라인 자동화
- 실험 결과를 기록, 관리할 때 해당 실험을 지칭하는 실험명 컨벤션 만들기
- 실험 결과 기록 시각화
    - WandB 사용 - 리드미에 기록할 때 시각적으로 확인 용이하다.

박재욱

나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

- 이번 프로젝트에서는 직접 딥러닝 모델을 실험해보고 적용하고자 하는 욕심이 있었다. 다행히 이것을 구현할 기회가 생겨서 여러 모델을 실험을 했지만 성능 향상에 도움이 되지 못해서 아쉬운 점이 있지만 실제로 그동안 어색했던 `torch` 모델을 사용했다는 점에서 의미있는 경험을 한 거 같아서 만족스럽다.

나는 어떤 방식으로 모델을 개선했는가?

- 최종 모델은 XGboost가 선정된 상황에서 다른 팀원들이 해당 모델에 도움이 되는 파생변수를 만드는 상황에서 나도 똑같이 파생변수를 생성하는 건 거시적인 관점에서 호율적이지 않다고 판단하여 경진대회에서 사용하는 모델 성능 올리는 것에 초점을 맞췄다. 이전 대회에서 모델의 일반화 성능이 아쉬워서 이를 어떻게 해결할 지에 대한 고민이 많았는데 찾아본 결과 일반화 성능을 올리는데 `seed ensemble` 이라는 방법이 존재했고 논리가 나의 니즈에 부합하여 적용해본 결과 성능 향상에 도움이 되었다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 이전 대회에서 협업 관점에서 많은 아쉬움을 남겼다. github을 사용하긴 했지만 이를 잘 사용한 지에 대한 의문이 남았다. 따라서 이를 해결하기 위해 Notion에 크게 `To-Do`, `In-Progress` 등 각자 하고 있는 작업 현황을 업데이트하여 하는 작업이 중복되지 않도록 하고 코드는 크게 모듈화를 진행하여 특정 기간까지 임무를 수행한 뒤 병합하는 과정을 거쳐 작업 효율성을 높였다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 위에서 언급했듯이 모델링에서 딥러닝 파트를 맡아서 여러가지 모델을 찾아보고 실제로 적용한 것에서 만족을 했지만 과연 경진대회와 같은 정형 데이터가 주어졌을 때 딥러닝 모델이 실질적으로 도움이 되는지에 대한 의문이 들었다. 물론 딥러닝을 사용해보고 싶어서 주도적으로 한 부분이 존재하지만 앞으로 경진대회에서 딥러닝 모델을 사용한다면 나에게 주어진 데이터를 고찰해보고 이에 적합한 모델을 선택하는게 트리 기반 모델을 실험하는 것보단 시간 낭비를 줄일 수 있다고 생각한다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 강의 중에 배운 모델의 파라미터에 따른 성능, 점수등 여러가지 지표를 대시보드로 표현해주는 wandb를 사용하지 못한 것이 아쉬웠다. 다음 프로젝트 때에는 이런 인프라 작업에 조금 더 주도적으로 초점을 맞추고 싶다.

서재은

Keep : 잘한 점

- 주도해서 Notion 페이지를 생성하여 팀원들이 진행상황을 기록할 수 있도록 관리했고, 깃허브 컨벤션이나 실험을 기록하도록 앞장섰다.

  결론적으로, 이전 프로젝트에 비해 협업 관리가 원활하게 이루어져 팀 단위에서도 만족스러웠고 결과도 좋게 나왔다.

- 주어진 데이터에 적절한 모델과 하이퍼파라미터(XGBoost + `lr`, `n_estimator`)를 찾아 대회 초반에 MAE 점수를 많이 낮출 수 있었다. 이 발견은 팀원들이 추후 실험들을 비교할 때 고정적으로 사용하는 기본 기준으로 사용했다.

- `torch` 라이브러리가 어색한 것이 항상 아쉬웠는데 모델을 직접 구현하며 많이 발전했다. 특히 `nn.Module`을 상속받아 직접 모델을 쌓은 점, `Dataset`과 `Dataloader`를 직접 커스텀해 우리의 데이터에 알맞은 형태로 변환할 수 있었다.

Problem : 한계 및 아쉬웠던 점

- 작업하다 놓치는 부분이 있어 오래 걸리는 전처리 단계를 여러번 반복한 적이 많았다. 조금 더 꼼꼼한 자세로 코딩해야겠다고 생각했다.
- 딥러닝 모델을 구현할 때 작성한 코드가 종속적이었다. 기존에 팀원들이 사용중이던 main 모듈에서 간편하게 딥러닝 모델을 불러오지 못해 아쉬웠다.

Try : 앞으로 시도해볼 점

- 개인적 바람으로 딥러닝 모델을 맡아 진행했는데 최종 모델링 성능 상승에 직접적인 기여를 하지 못했다. 다음 프로젝트를 진행하게 된다면 팀의 성능에 더 기여할 수 있게 앙상블 같은 기법을 사용해볼 것이다.

임태우

학습 목표

개인 학습 목표 다양한 feature 구현, 모델링 관련하여 단순 stacking ensemble이 아닌 다양한 앙상블 기법 구현과 하이퍼파라미터 최적화

공동 학습 목표 모듈화와 브랜치 전략을 통해 원활한 협업과 개인의 실험 결과를 공유하는 것

학습 목표 달성을 위한 노력

개인 학습 측면 다양한 EDA를 통해 데이터를 이해하고 새로운 feature 생성, OOF ensemble, K-Fold Cross Validation 구현, Optuna를 통한 하이퍼파라미터 최적화

공동 학습 측면 모듈화를 통한 우리 팀만의 베이스라인 코드 작성, Branch 전략과 Git 컨벤션 수립, 노션을 이용한 실험 결과 공유

나는 어떤 방식으로 모델을 개선했는가?

Feature Engineering

주어진 데이터를 기반으로 새로운 feature를 만드는데 집중했다. 초등학교, 중학교, 고등학교 별로 BallTree와 haversine을 이용해 각 학교까지의 최단 거리를 계산했고, 위도와 경도가 같은 중복 데이터가 많아 계산 시간이 오래 걸렸으나 중복 데이터를 제거하고 다시 매핑하여 시간을 단축했다. 또한 지하철역 데이터에서 중복이 발견되어, 중복 횟수별로 데이터를 지도에 표시해 환승역임을 확인했다. 단일역보다 환승역 주변의 집값이 높을 것으로 판단해 거리에 가중치를 부여하여 계산했다.

Modeling

Tableau 데이터에 좋은 성능을 보이는 트리 기반 모델을 중심으로 모델링을 진행했다. LightGBM, XGBoost, CatBoost 중 성능이 가장 좋았던 XGBoost 모델을 사용했고 Optuna로 하이퍼파라미터 최적화를 진행했으며, XGBoost 단일 모델로 10 K-fold 교차 검증을 적용해 public MAE와 validation MAE를 낮추고 일반화 성능을 높였다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

프로젝트의 target 값인 deposit을 feature engineering에 사용해도 된다는 팀의 논의에 따라 기존 위도와 경도를 기준으로 클러스터링한 함수에 타겟 인코딩 변수를 추가했다. 이는 permutation importance 기준으로 가장 높은 중요도를 보였지만, 실험 결과 기존의 top 20 변수를 사용했을 때보다 성능이 떨어졌으며 train 데이터에 과적합된 것으로 판단되어 아쉬움이 남았다. 또한 LightGBM과 XGBoost로 OOF 앙상블을 구현했으나, LightGBM의 성능이 좋지 않아 앙상블 결과 또한 개선되지 않았다. 이로 인해 앙상블이 항상 성능을 높이는 것은 아님을 깨달았으며, 더 많은 모델을 앙상블에 사용해보지 못한 점도 아쉽다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

앞선 두 프로젝트를 통해 머신러닝 모델을 충분히 다뤄본 만큼, 다음 프로젝트부터는 딥러닝 모델을 중심으로 적용해보고 싶다. 또한, 그동안 각자 실험 결과와 변수, 파라미터 등을 노션에 기록해 공유했지만, 이번에는 Wandb를 활용해 더 효율적으로 실험 결과를 관리하고 팀원들과 손쉽게 공유할 수 있도록 해볼 계획이다. 최종 프로젝트를 대비해 사용하고자 하는 기술 스택인 MLflow도 도입해보고 싶다. 마지막으로, 멘토님의 코드 리뷰 피드백을 반영해 코드 모듈화 작업에 기여하며 프로젝트의 완성도를 높이고자 한다.

최태순

잘했던 점

- EDA를 통해 다양한 feature engineering을 진행했다. 특히, deposit과 관련한 변수값을 추가하기 위한 고민을 많이 진행했다.
- 팀원들의 모든 파생변수를 하나로 합친 features.py 모듈화를 진행했다. 이를 통해 모델링 직전의 최종 데이터셋을 쉽게 구축할 수 있었다.
- xgboost와 feature importance를 통해 추출한 20개의 변수에서 높은 성능을 보여 최종 선택된 변수를 구할 수 있었다.

시도했으나 잘 되지 않았던 것들

- ensemble을 통해 구현했던 모델의 성능이 단일 모델보다 높은 성능을 보이지 못했다.
- deposit 관련 변수에서 높은 변수를 보였고 실질적으로 초반에 구축했던 변수보다 성능이 좋지 못했다.

아쉬웠던 점들

- 딥러닝 모델을 많이 시도해보지 못했던 점이 아쉽다.
- deposit 관련 변수를 초반에 미리 생성했다면 더 높은 성능을 끌어올릴 수 있었다고 생각하지만 이러지 못한 점이 아쉽다.
- 하이퍼파라미터 튜닝에 있어서 wandb를 활용하지 못했던 점이 아쉽다.

프로젝트를 통해 배운 점 및 시사점

- 팀원들과 하나의 프로젝트를 진행할 때 git의 merge와 branch 활용법을 습득할 수 있었다.
- 노션을 통해 각자 진행상황과 시도했던 변수, 모델 등을 공유함으로써 원활한 협업을 이어나갈 수 있었다.
- 또한, 이번 프로젝트의 핵심 중 하나인 모듈화를 집중적으로 진행했으며 이전의 프로젝트보다 더욱 깔끔하고 가독성 높은 코드를 구현할 수 있었다.

- 이전 프로젝트에서는 개별마다 시도했던 내용과 코드를 정리하지 못했지만 이러한 부분을 중심으로 보완함으로써 level2 proejct에서는 과정과 결과 모두 만족스러운 결과를 얻었다.

허진경

나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

이번 프로젝트에서 내 학습 목표는 **실험 내용과 결과를 명확하게 기록**하여 관리하고, **재현성을 확보**하는 것이었다. 또한 **data leakage에 유의함과 동시에 valid score에 대한 신뢰성을 확보**하고자 했다.

- 모델 실험 시마다, 사용한 피처와 모델, 하이퍼파라미터와 사용한 코드들을 포함해 해당 실험의 결과인 valid score와 public score를 모두 기록했다.
- 실험 재현성을 확보하기 위해 모두 동일한 랜덤 시드를 사용하고, 사용한 시드 번호를 저장해두었다.
- valid, test dataset의 내용이 학습 과정에 포함되지 않도록 clustering 과정과 같은 leakage가 발생할 수 있는 부분에서 train, valid, test dataset을 철저하게 분리하여 사용했다.

나는 어떤 방식으로 모델을 개선했는가?

feature engineering 부분에서는 두 가지 관점에서 새로운 피처를 추가하여 모델을 개선했다.

- 첫번째는 **타겟 자체의 위도, 경도에 대해 클러스터링을 하여 해당 군집에 대한 피처를 추가**했다. 클러스터링 결과에 따른 군집과, 포함되는 군집의 중심(centroid)과 데이터 샘플 간의 거리를 최종 피처로 추가했다.
- 두번째는 **타겟의 주변 환경에 대한 피처를 추가**했다. 타겟의 위치를 중심으로 반경 2km 이내의 공원 면적 총합, 학교 개수, 1km 반경 지하철역의 개수, 가장 가까운 지하철역까지의 거리 등을 추가했다.

modeling 부분에서는, 내가 개선을 시도할 때는 XGBoost에 중요도가 높은 상위 20개의 피처를 사용했을 때 성능이 괜찮다는 것이 확인된 상태였다. 그래서 해당 모델을 그대로 사용하되, 성능을 좀 더 끌어올리기 위해 데이터 샘플에 공간적 자기 상관이 존재할 것이라는 가설을 바탕으로 이를 반영해보고자 했다.

- 공간적 자기 상관을 반영하기 위해 **공간적 가중치 행렬을 생성**했다. 데이터셋의 크기가 너무 커서 이를 모두 메모리에 올릴 수 없기 때문에 데이터셋을 청크 단위로 분할하여 데이터 샘플과 나머지 데이터 샘플 간의 공간적 가중치를 계산하여 희소 행렬로 저장하는 방식으로 메모리를 관리하며 가중치 행렬을 생성했다.
- 공간적 가중치를 위도, 경도를 기준으로 생성했기 때문에 면적에 따른 전세가 차이를 고려하지 못한다는 부분을 생각해서 **면적 당 전세가**를 구한 뒤, 이를 **가중치 행렬과의 행렬곱을 통해 가중 평균을 구하여 피처로 추가**했다.

결과적으로, 기존의 모델에 공간적 가중치를 부여했을 때 성능이 좋아지는 부분에서 데이터에 종속변수에 대한 공간적 자기 상관이 존재할 가능성이 높다는 가설을 뒷받침할 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 공간적 행렬을 생성하는 과정에서 공간적 가중치 행렬이라고 해서 위도, 경도에 따른 거리만 고려하기보다는 built year 등과 같은 피처를 포함해서 계산했다면 좀 더 정확한 가중치 행렬이 만들어지지 않을까하는 아이디어를 떠올렸었다. 하지만 가중치 행렬을 생성하는 과정이 너무 오래걸렸기 때문에 시간적인 문제로 아이디어를 테스트해보거나 반영해보지 못했다.
- 데이터셋의 크기나 그에 따른 소요시간 등을 고려하지 못해서 모델이 돌아가는 동안 비는 시간을 잘 활용하지 못한 것이 아쉽다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇인가?

- 데이터셋의 크기가 클 때는, 새로운 아이디어나 개선점을 적용할 때 처음부터 전체 데이터에 적용하기 보다는 적당한 크기의 샘플을 뽑아 테스트해본 뒤 전체 데이터에 적용해야 할 것 같다.
- WandB, ML Flow같은 툴을 이용하여 실험 결과 기록을 좀 더 용이하게 해보고 싶다.
- 피처 하나하나가 모델 성능에 미치는 영향을 좀 더 기준을 명확히 정해 철저히 테스트하고 확인해보고 싶다.