Fast campus × upstage

Upstage AI Lab

Machine Learing Seminar | 2024. 11. 15(금)

목차

01

팀 소개

팀장/팀원 소개
협업 방식

* Team 사전오기 : 네번 쓰러져도 다섯번째 다시 일어날 것이다.

팀장

조성지

관심 분야 : 추천 시스템
전공 : 경영학과

역할 : 회의 진행 및 일정 관리

팀원

조혜인

관심 분야 : 데이터 엔지니어링
전공 : 컴퓨터공학

역할 : 외부 데이터 수집 및 분석

팀원

안서인

관심 분야 : NLP, 의료 도메인
전공 : 컴퓨터공학

역할 : 실험적인 시도 (데이터 증강 등)

팀원

김태환

관심 분야 : 에듀테크
전공 : 물리학

역할 : EDA, 데이터 시각화

경진대회 협업 방식

: Machine Learning [대회] Regression

협업 마인드셋 :

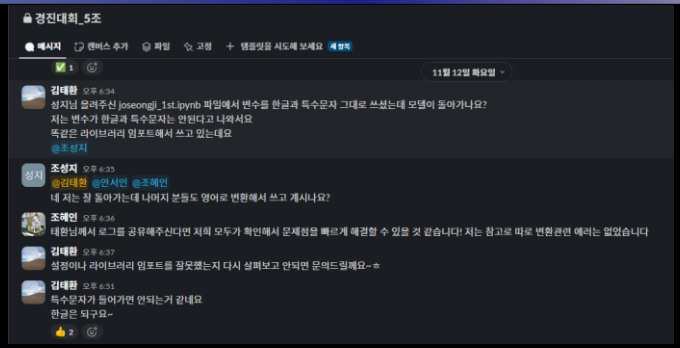
1) 1) 사소한 의견이라도 적극적으로 제시하기
2) 2) 새로 발견한 점이 있으면 공유하기
3) 3) 각자 직접 해보고 어려웠던 점 공유하기

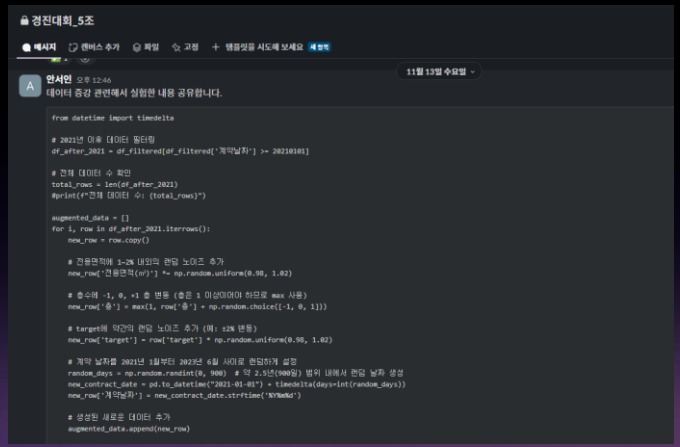
협업 진행 횟수 및 일정 : 매일 오전 출석 직후, 오후 퇴실 전. 소회의실에 모여 진행사항 공유

협업 진행 시 생긴 문제점 : 개인별 머신러닝 모델이 달라 결과 비교의 어려움

문제 해결 방법 : 모델의 다양성은 부족하지만 짧은기간 팀워크의 이점을 살리기 위해 모델과 하이퍼파라미터 통일

기타 : 코딩을 하며 발생한 문제점에 대해 실시간으로 공유해 문제 해결





02

경진대회 수행 절차 및 방법

목표 수립
수행 내용 / 수행 결과

경진대회 목표 수립

: House Price Prediction | 아파트 실거래가 예측

주제

House Price Prediction | 아파트 실거래가 예측

서울시 아파트 실거래가 매매 데이터를 기반으로 아파트 가격을 예측하는 대회

목표

정량적 목표 : RMSE 5000 이하 달성

정성적 목표 : ML 모델 개발 프로세스를 익히고
자신만의 개발 로직 세우기

개요

소개 및 배경 설명

부동산은 한국인의 삶에서 중요한 요소이다. 부동산 가격은 다양한 요인에 의하여 변동이 생기는데 부동산 실거래가 예측은 적절한 가격의 매매를 도와준다. 이러한 목적하에 다양한 부동산 관련 의사결정을 돕고자 부동산 실거래가 예측 모델을 개발하려 한다.

경진대회 목표는 정확하고 일반화된 모델을 개발하여 아파트 시장의 동향을 미리 예측하는 것이다.

기간

2024. 11. 04 ~ 2024. 11.14

경진대회 수행 내용

: House Price Prediction | 아파트 실거래가 예측

1
* 개발 환경 구축

NVIDIA 3090 서버 개인 할당

SSH 연결

VSCODE 작업

2
* 데이터 분석

입력피쳐와 데이터 샘플 수 확인

EDA를 통해 데이터 전반 파악

결측치 및 이상치 탐색 및 처리

연속형, 범주형 변수 처리

Feature Importance 기반 변수선택

3
* Feature 엔지니어링

EDA 와 도메인지식을 활용

지하철, 버스정류장 위치정보

연도별 분기별 금리 변화

데이터 증강

4
* 모델 선택 학습 및 평가

모델 : LightGBM

평가지표 : MSE, RMSE, MAE, R2

경진대회 수행 결과

: House Price Prediction | 아파트 실거래가 예측

데이터 기본정보 확인 및 결측치 파악

```
print(dt.columns)
print(f'훈련데이터 컬럼의 수는 {len(dt.columns)}입니다')
```

```
Index(['시군구', '번지', '본번', '부번', '아파트명', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도',
       '도로명', '해제사유발생일', '등기신청일자', '거래유형', '중개사소재지', 'k-단지분류(아파트,주상복합등등)',
       'k-전화번호', 'k-팩스번호', '단지소개기존clob', 'k-세대타입(분양형태)', 'k-관리방식', 'k-복도유형',
       'k-난방방식', 'k-전체동수', 'k-전체세대수', 'k-건설사(시공사)', 'k-시행사', 'k-사용검사일-사용승인일',
       'k-연면적', 'k-주거전용면적', 'k-관리비부과면적', 'k-전용면적별세대현황(60㎡이하)',
       'k-전용면적별세대현황(60㎡~85㎡이하)', 'k-85㎡~135㎡이하', 'k-135㎡초과', 'k-홈페이지',
       'k-등록일자', 'k-수정일자', '고용보험관리번호', '경비비관리형태', '세대전기계약방법', '청소비관리형태',
       '건축면적', '주차대수', '기타/의무/임대/임의=1/2/3/4', '단지승인일', '사용허가여부', '관리비 업로드',
       '좌표X', '좌표Y', '단지신청일', 'target'],
      dtype='object')
훈련데이터 컬럼의 수는 52입니다
```

```
print(dt_test.columns)
print(f'테스트데이터 컬럼의 수는 {len(dt_test.columns)}입니다')
```

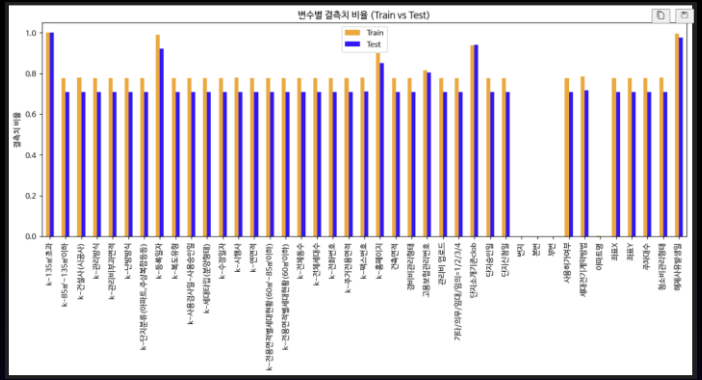
```
Index(['시군구', '번지', '본번', '부번', '아파트명', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도',
       '도로명', '해제사유발생일', '등기신청일자', '거래유형', '중개사소재지', 'k-단지분류(아파트,주상복합등등)',
       'k-전화번호', 'k-팩스번호', '단지소개기존clob', 'k-세대타입(분양형태)', 'k-관리방식', 'k-복도유형',
       'k-난방방식', 'k-전체동수', 'k-전체세대수', 'k-건설사(시공사)', 'k-시행사', 'k-사용검사일-사용승인일',
       'k-연면적', 'k-주거전용면적', 'k-관리비부과면적', 'k-전용면적별세대현황(60㎡이하)',
       'k-전용면적별세대현황(60㎡~85㎡이하)', 'k-85㎡~135㎡이하', 'k-135㎡초과', 'k-홈페이지',
       'k-등록일자', 'k-수정일자', '고용보험관리번호', '경비비관리형태', '세대전기계약방법', '청소비관리형태',
       '건축면적', '주차대수', '기타/의무/임대/임의=1/2/3/4', '단지승인일', '사용허가여부', '관리비 업로드',
       '좌표X', '좌표Y', '단지신청일'],
      dtype='object')
테스트데이터 컬럼의 수는 51입니다
```

```
# 결측치가 없는 열만 선택
no_missing_columns_train = dt.columns[dt.isnull().sum() == 0]
print("훈련데이터 결측치가 없는 변수:")
print(no_missing_columns_train)
no_missing_columns_test = dt_test.columns[dt_test.isnull().sum() == 0]
print("테스트데이터 결측치가 없는 변수:")
print(no_missing_columns_test)
```

```
훈련데이터 결측치가 없는 변수:
Index(['시군구', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도', '도로명', '등기신청일자', '거래유형',
       '중개사소재지', 'target'],
      dtype='object')
테스트데이터 결측치가 없는 변수:
Index(['시군구', '본번', '부번', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도', '도로명',
       '등기신청일자', '거래유형', '중개사소재지'],
      dtype='object')
```

```
dt_test.columns
```

```
Index(['시군구', '번지', '본번', '부번', '아파트명', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도',
       '도로명', '해제사유발생일', '등기신청일자', '거래유형', '중개사소재지', 'k-단지분류(아파트,주상복합등등)',
       'k-전화번호', 'k-팩스번호', '단지소개기존clob', 'k-세대타입(분양형태)', 'k-관리방식', 'k-복도유형',
       'k-난방방식', 'k-전체동수', 'k-전체세대수', 'k-건설사(시공사)', 'k-시행사', 'k-사용검사일-사용승인일',
       'k-연면적', 'k-주거전용면적', 'k-관리비부과면적', 'k-전용면적별세대현황(60㎡이하)',
       'k-전용면적별세대현황(60㎡~85㎡이하)', 'k-85㎡~135㎡이하', 'k-135㎡초과', 'k-홈페이지',
       'k-등록일자', 'k-수정일자', '고용보험관리번호', '경비비관리형태', '세대전기계약방법', '청소비관리형태',
       '건축면적', '주차대수', '기타/의무/임대/임의=1/2/3/4', '단지승인일', '사용허가여부', '관리비 업로드',
       '좌표X', '좌표Y', '단지신청일', 'target'],
      dtype='object')
```

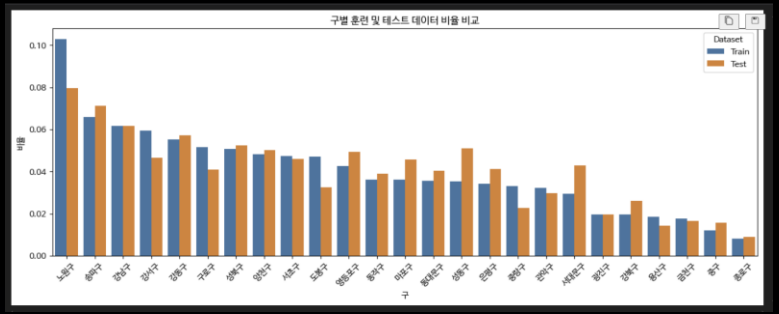


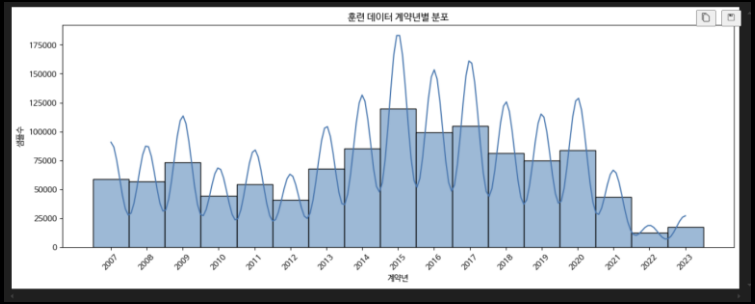
훈련 데이터 : 1118822 개의 행과 52개 컬럼
테스트 데이터 : 9272 개의 행과 51 개 컬럼
결측치비율 : 결측치가 없는 컬럼이 10개

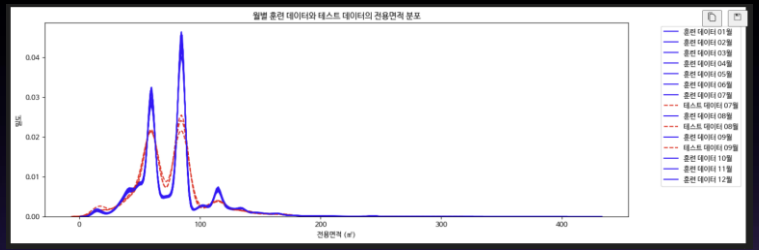
경진대회 수행 결과

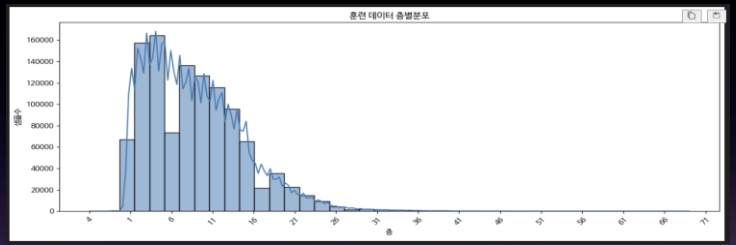
: House Price Prediction | 아파트 실거래가 예측

EDA 진행 (입력변수)







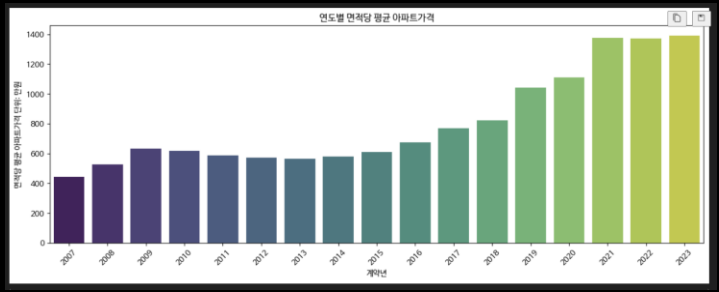


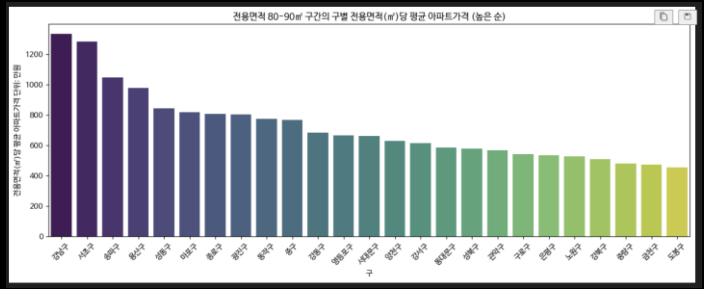
구별, 계약년별, 전용면적별, 층수 별 데이터 분포를 살펴봄

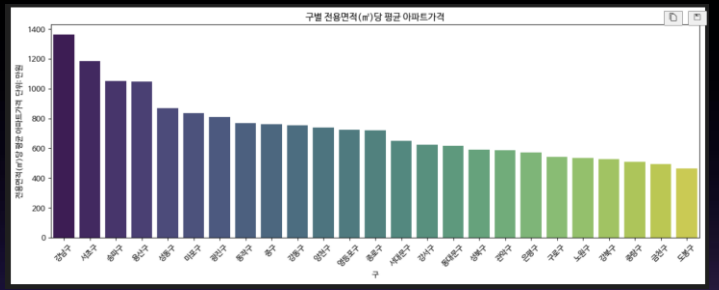
경진대회 수행 결과

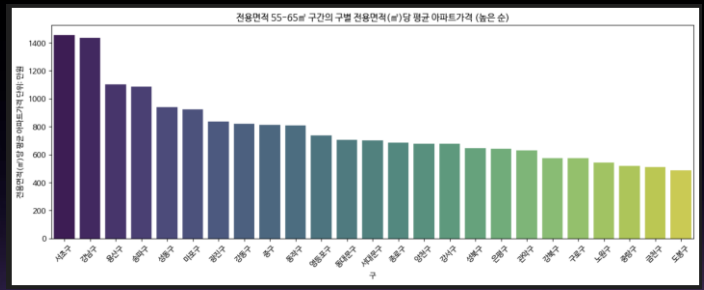
: House Price Prediction | 아파트 실거래가 예측

EDA 진행 (Target)









타겟의 가격분포를 연도별, 구별, 구별전용면적 으로 살펴봄

경진대회 수행 결과

: House Price Prediction | 아파트 실거래가 예측

결측치 처리

```
for column in dt.columns:
    if dt.dtypes[column] == 'object':
        dt[column] = dt[column].astype('category')

for column in dt_test.columns:
    if dt_test.dtypes[column] == 'object':
        dt_test[column] = dt_test[column].astype('category')
```

```
df_filtered = dt[['시군구', '번지', '본번', '부번', '아파트명', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도', '도로명']]
df_erased = dt[list(set(dt.columns) - set(df_filtered.columns))]
```

```
df_filtered['계약년월'] = df_filtered['계약년월'].astype('str')
df_filtered['계약일'] = df_filtered['계약일'].astype('str').apply(lambda x: x.zfill(2))
df_filtered['계약날짜'] = df_filtered['계약년월'] + df_filtered['계약일']
df_filtered['계약날짜'] = df_filtered['계약날짜'].astype('int')
df_filtered = df_filtered.drop(columns=['계약년월', '계약일'])

```

```
df_test = dt_test[['시군구', '번지', '본번', '아파트명', '전용면적(㎡)', '계약년월', '계약일', '층', '건축년도', '도로명']]
df_test_erased = dt_test[list(set(dt_test.columns) - set(df_test.columns))]
```

```
df_test['계약년월'] = df_test['계약년월'].astype('str')
df_test['계약일'] = df_test['계약일'].astype('str').apply(lambda x: x.zfill(2))
df_test['계약날짜'] = df_test['계약년월'] + df_test['계약일']
df_test['계약날짜'] = df_test['계약날짜'].astype('int')
df_test = df_test.drop(columns=['계약년월', '계약일'])
```

```
# 랜덤 포레스트 모델 훈련
rf = RandomForestClassifier(n_estimators=50,
                            random_state=42,
                            n_jobs=-1,
                            max_depth=20,
                            min_samples_split=10,
                            min_samples_leaf=5,
                            warm_start=False)
rf.fit(X_train, y_train)

val_accuracy = rf.score(X_val, y_val)
print(f"Validation Accuracy: {val_accuracy:.4f}")

# 결측치 예측 준비
X_missing = no_aparment_name[features]
for col in ['시', '군', '구', '번지']:
    X_missing[col] = le.transform(X_missing[col].astype(str))
X_missing = pd.DataFrame(imputer.transform(X_missing), columns=X_missing.columns)

# 결측치 예측
predicted_names_encoded = rf.predict(X_missing)

# 예측된 값을 원래 레이블로 변환
predicted_names = le.inverse_transform(predicted_names_encoded)

# 결측치 채우기
concat_data.loc[concat_data['아파트명'].isna(), '아파트명'] = predicted_names

# 결과 확인
print("남은 결측치 수:", concat_data['아파트명'].isnull().sum())

print("₩n예측된 아파트명 상위 10개:")
print(concat_data.loc[concat_data.index.isin(no_aparment_name.index), '아파트명'].value_counts().head(10))
```

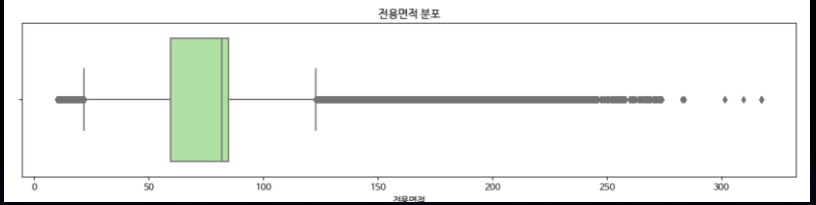
도메인 지식 기반, 학습기여도가 적다고 판단한 일부열은 제거하고 RandomForest모델을 통해 결측치 보간

경진대회 수행 결과

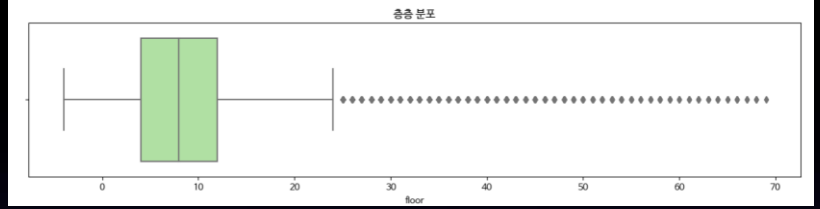
: House Price Prediction | 아파트 실거래가 예측

이상치제거

```
# 대표적인 연속형 변수인 "전용 면적" 변수 관련한 분포
fig = plt.figure(figsize=(15, 3))
sns.boxplot(data = concat_select, x = '전용면적(㎡)', color='lightgreen')
plt.title('전용면적 분포')
plt.xlabel('전용면적')
plt.show()
```




```
# 연속형 변수인 "층" 변수 관련한 분포
fig = plt.figure(figsize=(15, 3))
sns.boxplot(data = concat_select, x = '층', color='lightgreen')
plt.title('층 분포')
plt.xlabel('floor')
plt.show()
```

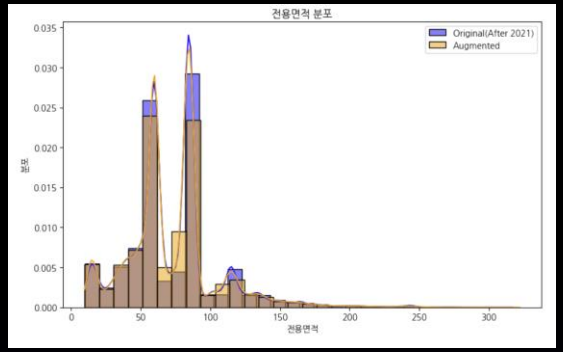


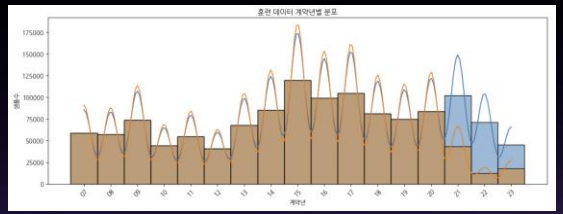
IQR을 이용하여 이상치를 탐지하고 컬럼별로 분석하여 삭제 및 대체 혹은 유지

경진대회 수행 결과

: House Price Prediction | 아파트 실거래가 예측

데이터증강





```python
from datetime import timedelta

# 2021년 이후 데이터 필터링
df_after_2021 = df_filtered[df_filtered['계약년월'] >= 202101]

# 전체 데이터 수 확인
total_rows = len(df_after_2021)
#print(f"전체 데이터 수: {total_rows}")

augmented_data = []
for _ in range(2) :
    for i, row in df_after_2021.iterrows():
        new_row = row.copy()

        # 전용면적에 4% 내외의 랜덤 노이즈 추가
        new_row['전용면적(㎡)'] *= np.random.uniform(0.96, 1.04)

        # 층수에 -1, 0, +1 층 변동 (층은 1 이상이어야 하므로 max 사용)
        new_row['층'] = max(1, row['층'] + np.random.choice([-1, 0, 1]))

        # target에 약간의 랜덤 노이즈 추가 (예: ±5% 변동)
        new_row['target'] = row['target'] * np.random.uniform(0.96, 1.04)

        # 계약 날짜를 2021년 1월부터 2023년 6월 사이로 랜덤하게 설정
        random_days = np.random.randint(0, 900)  # 약 2.5년(900일) 범위 내에서 랜덤 날짜 생성
        new_contract_date = pd.to_datetime("2021-01") + timedelta(days=int(random_days))
        new_row['계약년월'] = new_contract_date.strftime('%Y%m')

        # 생성된 새로운 데이터 추가
        augmented_data.append(new_row)


# 새로운 데이터프레임 생성
df_augmented = pd.DataFrame(augmented_data)
```

타겟에 가까운 연도의 데이터 부족을 보완하기 위하여 데이터 증강을 함

경진대회 수행 결과

수행 결과

: House Price Prediction | 아파트 실거래가 예측

파생변수 생성

```
date_transaction = features.groupby('계약날짜').count().reset_index()[['계약날짜', '아파트명']]
date_transaction = date_transaction.rename(columns={'아파트명': '거래량'})
date_transaction = date_transaction.sort_values('계약날짜')

date_transaction_test = test.groupby('계약날짜').count().reset_index()[['계약날짜', '아파트명']]
date_transaction_test = date_transaction_test.rename(columns={'아파트명': '거래량'})
date_transaction_test = date_transaction_test.sort_values('계약날짜')

all_transaction = pd.concat([date_transaction, date_transaction_test])
all_transaction['7days_avg_transaction'] = all_transaction['거래량'].rolling(window=7, min_periods=1).mean()
all_transaction['14days_avg_transaction'] = all_transaction['거래량'].rolling(window=14, min_periods=1).mean()
all_transaction['30days_avg_transaction'] = all_transaction['거래량'].rolling(window=30, min_periods=1).mean()
all_transaction['60days_avg_transaction'] = all_transaction['거래량'].rolling(window=60, min_periods=1).mean()
all_transaction['120days_avg_transaction'] = all_transaction['거래량'].rolling(window=120, min_periods=1).mean()
all_transaction['360days_avg_transaction'] = all_transaction['거래량'].rolling(window=360, min_periods=1).mean()
all_transaction['720days_avg_transaction'] = all_transaction['거래량'].rolling(window=720, min_periods=1).mean()
```

```
all_transaction['7days_std'] = all_transaction['거래량'].rolling(window=7, min_periods=1).std()
all_transaction['14days_std'] = all_transaction['거래량'].rolling(window=14, min_periods=1).std()
all_transaction['30days_std'] = all_transaction['거래량'].rolling(window=30, min_periods=1).std()
all_transaction['60days_std'] = all_transaction['거래량'].rolling(window=60, min_periods=1).std()
all_transaction['120days_std'] = all_transaction['거래량'].rolling(window=120, min_periods=1).std()
all_transaction['360days_std'] = all_transaction['거래량'].rolling(window=360, min_periods=1).std()
all_transaction['720days_std'] = all_transaction['거래량'].rolling(window=720, min_periods=1).std()
```

```
all_transaction['7days_std_mean'] = all_transaction['7days_std'] / all_transaction['7days_avg_transaction']
all_transaction['14days_std_mean'] = all_transaction['14days_std'] / all_transaction['14days_avg_transaction']
all_transaction['30days_std_mean'] = all_transaction['30days_std'] / all_transaction['30days_avg_transaction']
all_transaction['60days_std_mean'] = all_transaction['60days_std'] / all_transaction['60days_avg_transaction']
all_transaction['120days_std_mean'] = all_transaction['120days_std'] / all_transaction['120days_avg_transaction']
all_transaction['360days_std_mean'] = all_transaction['360days_std'] / all_transaction['360days_avg_transaction']
all_transaction['720days_std_mean'] = all_transaction['720days_std'] / all_transaction['720days_avg_transaction']
```

```
# 구별로 샘플수가 다른만큼 비율에 대한 파생변수 생성
concat_select['구별비율'] = concat_select['구'].map(concat_select['구'].value_counts(normalize=True))
```

```
# 연도별 데이터 비율이 일정하지 않아 비율지수 생성
concat_select['년도별_비율'] = concat_select['계약년'].map(concat_select['계약년'].value_counts(normalize=True))
```

```
# 3층에서 16층 사이가 많은 샘플을 차지하므로 파생변수 생성
concat_select['층_3to16'] = concat_select['층'].apply(lambda x: 1 if 3 <= x <= 16 else 0)
```

```
# 2015년을 기점으로 가격변동가 크게 나뉘므로 계약년이 2015 이전은 0, 2015 이후는 1로 설정하는 파생변수 생성
concat_select['계약년_구간'] = concat_select['계약년'].apply(lambda x: 0 if x < 2015 else 1)
```

```
# 특정 면적대가 많은 비율을 차지하므로
# 전용면적 구간에 따른 레이블 변수 생성
def area_label(area):
    if 55 <= area <= 65:
        return 2
    elif 80 <= area <= 90:
        return 3
    elif 110 <= area <= 120:
        return 1
    else:
        return 0  # 해당 구간이 아닐 경우 0으로 설정

concat_select['전용면적_구간'] = concat_select['전용면적(㎡)'].apply(area_label)
```

```
# 계약년도, 계약월 추가

df_train['near_station1'] = df_train['distance'] < 500
df_test['near_station1'] = df_test['distance'] < 500
df_train['near_station2'] = df_train['distance'] < 1000
df_test['near_station2'] = df_test['distance'] < 1000
df_train['구'] = df_train['시군구'].str.split().str[1]
df_test['구'] = df_test['시군구'].str.split().str[1]
df_train['동'] = df_train['시군구'].str.split().str[2]
df_test['동'] = df_test['시군구'].str.split().str[2]
df_train['계약년도'] = df_train['계약날짜'].astype('str').str[:4]
df_test['계약년도'] = df_test['계약날짜'].astype('str').str[:4]
df_train['계약월'] = df_train['계약날짜'].astype('str').str[4:6]
df_test['계약월'] = df_test['계약날짜'].astype('str').str[4:6]
```

	시군구	번지	본번	부번	아파트명	전용면적(㎡)	층	건축년도	도로명	station_x	계약날짜
0	서울특별시 강남구 개포동	658-1	658.0	1.0	개포6차우성	79.9700	5	1987	언주로 3	구룡역	20230726
1	서울특별시 강남구 개포동	651-1	651.0	1.0	개포더샵트리에	108.2017	10	2021	개포로 311	구룡역	20230815
2	서울특별시 강남구 개포동	652	652.0	0.0	개포우성3차	161.0000	15	1984	개포로 307	구룡역	20230728
3	서울특별시 강남구 개포동	652	652.0	0.0	개포우성3차	133.4600	14	1984	개포로 307	구룡역	20230810
4	서울특별시 강남구 개포동	652	652.0	0.0	개포우성3차	104.4300	6	1984	개포로 307	구룡역	20230818

```
df_filtered.head()
```

	시군구	번지	본번	부번	아파트명	전용면적(㎡)	층	건축년도	도로명	station_x	target	계약날짜
0	서울특별시 강남구 개포동	658-1	658.0	1.0	개포6차우성	79.97	3	1987	언주로 3	구룡역	124000	20171208
1	서울특별시 강남구 개포동	658-1	658.0	1.0	개포6차우성	79.97	4	1987	언주로 3	구룡역	123500	20171222
2	서울특별시 강남구 개포동	658-1	658.0	1.0	개포6차우성	54.98	5	1987	언주로 3	구룡역	91500	20171228
3	서울특별시 강남구 개포동	658-1	658.0	1.0	개포6차우성	79.97	4	1987	언주로 3	구룡역	130000	20180103
4	서울특별시 강남구 개포동	658-1	658.0	1.0	개포6차우성	79.97	2	1987	언주로 3	구룡역	117000	20180108

EDA와 도메인 지식에 기반하여 여러 파생변수 생성 (지하철역, 거래량, 면적별분포 등)

경진대회 수행 결과

: House Price Prediction | 아파트 실거래가 예측

모델훈련

```python
# 파라미터 그리드 설정
param_grid = {
    'num_leaves': [20, 31],
    'learning_rate': [0.05, 0.1, 0.15],
    'n_estimators': [100, 500],
    'max_depth': [7, 15],
    'min_child_samples': [10, 20],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# ParameterGrid를 이용해 모든 파라미터 조합 생성
param_combinations = list(ParameterGrid(param_grid))
total_combinations = len(param_combinations)

best_score = float('inf')
best_params = None

# tqdm을 이용하여 진행 상황 출력
for i, params in enumerate(tqdm(param_combinations, total=total_combinations, desc="Grid Search Progress")):
    model = lgb.LGBMRegressor(**params, verbose=-1)
    model.fit(x_train, y_train)
    y_train_pred = model.predict(x_train)
    y_pred = model.predict(x_valid)
    score = mean_squared_error(y_valid, y_pred, squared=False)

    # 최적 파라미터 갱신
    if score < best_score:
        best_score = score
        best_params = params
```

```python
model = lightgbm.LGBMRegressor(
    boosting_type='gbdt',      # 기본 부스팅 방식 (gbdt는 그레디언트 부스팅 결정 트리)
    objective='regression',    # 회귀 문제를 해결하기 위한 목적 함수 설정
    num_leaves=31,             # 리프 노드의 최대 개수
    learning_rate=0.15,        # 학습률 (작을수록 학습 속도가 느려짐, 과적합 방지)
    n_estimators=1000,         # 부스팅 반복 횟수 (트리 개수)
    max_depth=15,              # 트리의 최대 깊이 (-1은 제한 없음)
    min_child_samples=20,      # 리프 노드의 최소 데이터 수 (과적합 방지)
    subsample=0.6,             # 각 트리 학습 시 사용할 샘플 비율 (0.8은 80%만 사용)
    colsample_bytree=0.6,      # 각 트리 학습 시 사용할 특징 비율 (0.8은 80%만 사용)
    reg_alpha=0.1,             # L1 정규화 (과적합 방지)
    reg_lambda=0.1             # L2 정규화 (과적합 방지)
)
```

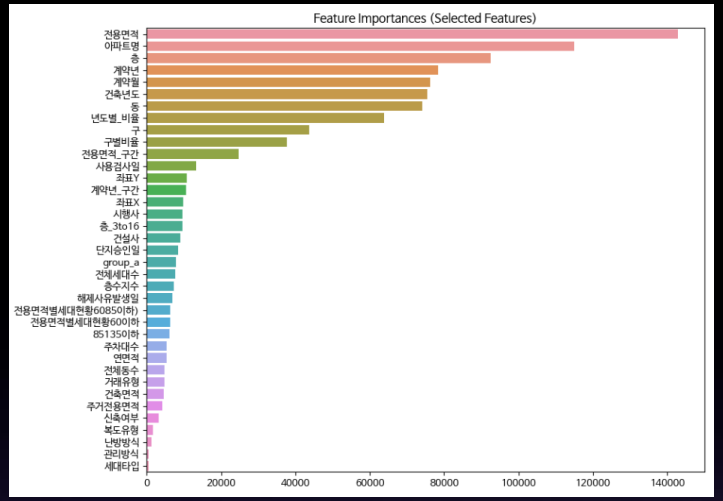
```python
# 01번 모델 RMSE: 4399.69375663948
model.fit(x_train, y_train)
```

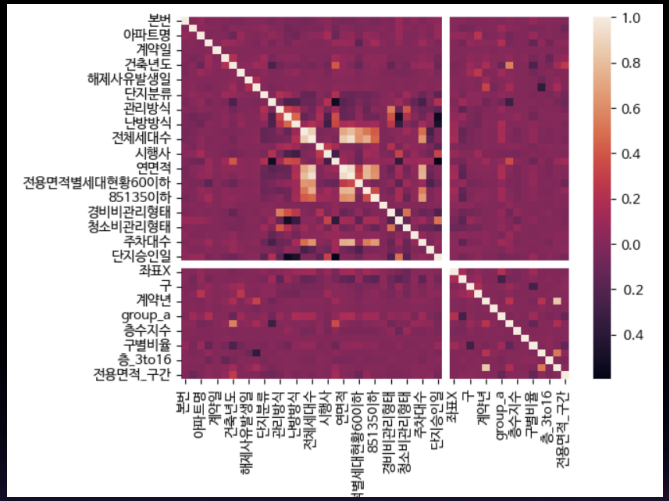
Grid Search 와 평가를 통해 데이터 규모에 맞는 LightGBM 모델의 최적의 파라미터 탐색

경진대회 수행 결과

: House Price Prediction | 아파트 실거래가 예측

Feature Importance & Correlation Aanalysis





컬럼을 최대한 반영한 모델의 경우 Feature Importance 와 상관관계 분석을 통해 변수선택

03

분석 인사이트 및 결과

문제 및 인사이트 도출
해결 방법 및 결과

경진대회 인사이트 공유

: House Price Prediction | 아파트 실거래가 예측

01. 문제 발생 배경 및 원인 분석

결측치가 범위가 넓음

전체칼럼의 80%가 결측치를 가짐

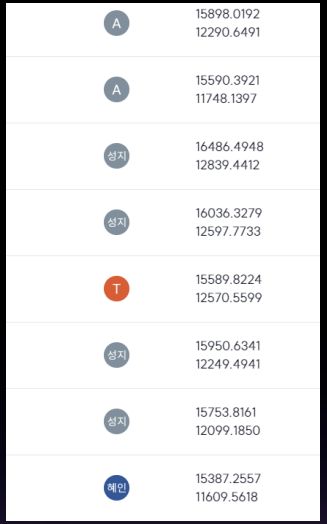
결측치를 가진 칼럼의 결측치 비율이 80% 이상임

중요도가 낮은 칼럼을 제거하고 모델을 훈련

파라미터 튜닝을 통해서 예측 성능을 향상시킴

모델파라미터 튜닝을 통한 성능향상의 한계에 부딪침

모델이 의미있게 분류를 할 수 있는 추가 변수가 없다고 판단됨



경진대회 인사이트 공유

: House Price Prediction | 아파트 실거래가 예측

02. 인사이트 도출

모델에 추가 시계열 정보 제공

금리변화를 데이터에 추가

지하철역의 가중치 추가 부여

추가 데이터 증강

딥러닝모델을 통한 성능향상

경진대회 인사이트 공유

: House Price Prediction | 아파트 실거래가 예측

03. 해결방법

추가 시계열 정보
-〉 거래량의 7,14,30,60,90 이동평균선 컬럼생성

금리변화 반영 파생변수 생성
-〉 모델링중 시간 부족으로 테스트 진행못함

지하철역 추가 가중치 부여
-〉 주요 상업지구나 주거지 밀집구역, 신축 아파트가 많은 지하철에 가중치

04. 결과

유의미한 성능 향상은 없었음

-〉 도메인 기반하여 유용한 외부자료를 찾아 데이터의 패턴을 반영할 수 있는 파생변수를 생성해야 할 것으로 판단됨

15849.6977 11930.6826
16028.0830 12311.1032
15626.9619 12083.9796

04

회고

우리 팀의 목표 달성도
느낀점 및 향후 계획

경진대회 회고

: House Price Prediction | 아파트 실거래가 예측

Point 1 우리 팀의 처음 목표에서 어디까지 도달했는가

정량적 목표 : RMSE 5000 이하 달성

정량적 목표 평가 : RMSE (최종) 점수 -> 11809.62
목표대비 달성률 -> 42%

Point 2 우리 팀이 잘했던 점

파라미터 튜닝 최적화된 단일 ML 모델 사용으로 시간절약 및 다른 실험들에 집중할 수 있었음

슬랙방과 줌채팅방을 통한 실시간 정보공유와 소통으로 모델링중 발생한 문제들을 해결할 수 있었음

Point 3 협업하면서 아쉬웠던 점

아쉬운 점 : 시간 효율적인 측면에서 한가지 머신러닝 모델을 사용했는데 다양한 모델을 시도하지 못한 점

향후 계획 : 향후 팀워크가 중요한 프로젝트 수행에 있어서는 명확한 역할 분담을 통해 팀과 개인의 유기적인 결합을 시도

경진대회 진행 소감

: House Price Prediction | 아파트 실거래가 예측

느낀점

*** 조성지**

첫 프로젝트라 방향성을 잡지 못한 점이 조금 아쉬웠습니다.
도메인 지식에 대한 고민 비중을 높여야 겠다고 생각했습니다.

*** 조혜인**

다음에는 더욱 열심히 해서 좋은 결과물을 만들어 내고 싶습니다.
생각이 너무 많아서 중간과정이 입 밖으로 잘 안 나오는데,
생각을 정리하는 연습을 해서 더 많은 의견을 제출 할 수 있도록 노력하겠습니다.

*** 안서인**

생각한 것을 코드로 바로 구현하지 못하는 것이 저의 부족한 점이라고 생각했는데
이번 스터디를 통해 코드를 짜는 것에 조금이나마 익숙해진 것 같습니다.
앞으로도 생각한 것을 결과로 구현하기 위해 고민하는 시간을 갖도록 노력하겠습니다.

*** 김태환**

프로젝트 주제에 대한 탐색을 충분히 한 다음 프로세스를 세우는게 중요함을 느꼈습니다.
다음에는 가설과 실험을 검증할 수 있는 효율적인 코드 구현을 하고 싶습니다.

Life-Changing Education

감사합니다.