■ 서식 2 : 결과보고서 요약본

Project Based Learning 결과보고서 요약본

<table>
<tr><td rowspan="2">교과목명</td><td>국문</td><td colspan="3">인공지능과데이터과학(2)</td></tr>
<tr><td>영문</td><td colspan="3">AI and Data Science(2)</td></tr>
<tr><td colspan="2">PBL 관련 능력단위
(능력단위코드)</td><td colspan="3">PBL 관련 능력단위요소 (능력단위요소코드)</td></tr>
<tr><td colspan="2" rowspan="3">데이터 시각화 기술, 모델링과 예측,
텍스트 마이닝 이해하기
(YCS-2001020907_20v1)</td><td colspan="3">데이터 과학 방법론 이해하기</td></tr>
<tr><td colspan="3">회귀 분석, 텍스트 분석 이해하기</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="2">학년 반</td><td>2학년 1반</td><td>조원</td><td>능지공인 팀</td></tr>
<tr><td colspan="2">프로젝트 주제</td><td colspan="3">서울시 부동산 실거래가 데이터셋에 머신러닝 기반 회귀분석을 수행하여 건물 가격에 영향을 미치는 항목을 확인하고 그에 따른 건물가격을 예측한다.</td></tr>
<tr><td colspan="2">지도교수</td><td>금 득 규</td><td>산업체 참가여부</td><td>☐ 유 ☑ 무</td></tr>
<tr><td colspan="2">참여학생</td><td colspan="3">이은우</td></tr>
<tr><td colspan="2">작품 개요
(주제 선정 이유)</td><td colspan="3">회귀분석의 기본적인 예시인 건물의 특징인 건물면적, 건물의 층수, 건축년도등을 이용하여 건물의 가격을 예측하는 문제를 선정함.
[데이터 셋은 서울시 부동산 실거래 자료를 사용]</td></tr>
<tr><td colspan="2">작품 구조도
(문제점 제시 및 개선방안)</td><td colspan="3">회귀분석은 변수 간의 관계를 분석하고, 한 변수(Y : 종속변수 = 건물의 가격)를 다른 변수들(X : 독립변수들 = 건물의 특징)을 이용해 예측하는 통계 기법.
1) 먼저 이 회귀분석 주제에 알맞은 적절한 데이터셋을 구해야함.
▸공공데이터 포털 -> 서울열린데이터광장 -> 서울시 부동산 실거래가 정보.csv
2) 회귀분석 이전에 데이터를 잘 다듬는 데이터 전처리하는 과정을 거친다.
이 과정이 있어야 엉뚱한 분석결과를 피할 수 있고, 원하는 분석 결과를 얻을 수 있기 때문.
▸ 데이터 필터링 : 구역이 너무 많은 관계로 일부분의 구역만 불러옴[개봉동, 오류동, 온수동의 데이터값만 걸러냄]
▸결측값 처리 : 데이터 분석시 누락되거나 잘못 입력된 데이터셋의 값. 즉, 특징의 데이터값이 NULL값인 경우 제거 및 처리함.
▸원 핫 인코딩 : 건물의 용도와 법정동명은 숫자로 표시가 불가능한 범주형 데이터로 원 핫 인코딩이라는 단어를 표현하는 가장 기본적인 방법을 이용함. 하지만 이 프로젝트에서 범주형 데이터를 처리하기에 어려움이 있어 제외시킴.
3) 독립변수(X)와 종속변수(Y)를 지정후 선형회귀 분석 모델 객체를 생성하고 훈련데이터를 이용해 학습시킴.
▸ 평균제곱 오차(MSE)와 결정 계수(R^2)를 이용해 실제값과 예측값 사이의 오차, 종속변수의 분산을 알아냄
▸ 이때 처음 생성한 모델로 학습을 시킨 결과 오차가 백분율로 표현했을 때 34%였으며 이를 줄이기 위해 여러 가지 개선방안을 찾음.</td></tr>
</table>

A. 데이터 전처리 과정

▸**파생 변수 생성** : 기존 데이터의 변수를 활용해 새로운 정보를 생성하는 변수로, 모델이 중요한 관계를 더 잘 학습하도록 돕기 위해 사용, 예를 들어 건축년도 보다 건축된 시점이 오래된 정도[현재 년도 - 건축 년도]를 직관적으로 나타내어 모델 성능과 해석력을 향상시킴.

▸**로그 변환** : 건물 가격의 분포가 큰 관계로 프로그램 계산값이 inf가 나오게 된다. 즉, 변수를 측정하는 단위가 매우 크거나, 매우 작아서 생기는 문제를 정규에 가깝게 로그 변환을 이용하여 해결한다.

B. 모델 설계 과정

▸ **데이터 스케일링** : StandardScaler를 사용해 데이터의 스케일을 맞춤으로서 모델이 독립변수의 범위 차이로 인해 잘못된 학습을 하는 것을 방지함.

▸ **하이퍼 패러미터 튜닝** : 모델의 성능향상을 위해 쓰이는 기법중 하나인 GridSearchCV를 기반으로 한 RandomizedSearchCV()라는 모듈을 사용하여 모델의 하이퍼 패러미터의 값을 리스트로 입력해 값에 대한 경우의 수마다 예측 성능을 측정 평가해 비교하며 최적의 하이퍼 패러미터 값을 찾는 과정을 진행.

이렇게 하여 하이퍼파라미터 검색 결과 가장 좋은 조합으로 설정된 모델을 반환하고, 최적화된 모델을 훈련데이터로 학습을 시키는 과정을 진행함.

4)훈련 후 회귀분석 결과인 실제값과 예측값의 차이를 시각화로 나타내어 비교함.

실제값(테스트 데이터의 원래 값)과 예측값(모델의 예측값)의 분포를 히스토그램으로 나타내 실제값과 예측값이 어떤 분포를 가지며, 두 분포가 얼마나 유사한지를 시각적으로 확인하고, 테스트 데이터의 각 특징[독립변수로 채택된 것들]에 대해 실제값과 예측값의 관계를 “산점도”로 시각화하고, 모델이 데이터를 얼마나 잘 예측했는지를 평가함.

5)건물의 특징을 임의의 데이터 값을 넣어보고 그 데이터값을 토대로 건물 가격을 예측함.

관련 이론

회귀분석, 선형회귀, 데이터 전처리 이론, 머신러닝 알고리즘[랜덤포레스트], 하이퍼 파라미터 튜닝, 사이킷런 라이브러리

결과물 제작 (문제점 개선사항)

★1. 데이터 준비

```
import pandas as pd
import numpy as np
data = pd.read_csv('/content/drive/MyDrive/데이터 과학/PBL/서울시 부동산 실거래가 정보(r2).csv', encoding='cp949')
data.head()
```

자치구명	법정동코드	법정동명	지번구분	지번구분명	본번	부번	건물명	...	물건금액(만원)	건물면적(㎡)	토지면적(㎡)	층
구로구	10600	고척동	1.0	대지	306.0	0.0	경남2	...	53000	70.980	0.0	5.0
구로구	10600	고척동	1.0	대지	306.0	0.0	경남2	...	53000	70.980	0.0	5.0
구로구	10200	구로동	1.0	대지	104.0	9.0	성호메이플라워멤버스빌	...	15000	31.750	0.0	6.0
구로구	10900	궁동	1.0	대지	290.0	0.0	우남푸르미아	...	63000	84.958	0.0	7.0

데이터를 불러오고 처리하기 위해 pandas와 numpy 라이브러리를 불러온다. 이번 회귀 분석의 메인 데이터셋인 “서울시 부동산 실거래가 정보” csv파일을 불러온다. 그리고 head함수를 통해 불러온 데이터를 확인한다.

★2. 데이터 전처리

```
# 필요한 열 선택
columns = ['물건금액(만원)', '건물면적(㎡)', '층', '건축년도', '법정동명']
data_selected = data[columns]

# 법정동명이 '오류동', '온수동'인 데이터만 필터링
filtered_data = data_selected[data_selected['법정동명'].isin(['오류동', '온수동'])].copy()

# 결측값 처리
data_cleaned = filtered_data.dropna(subset=['층', '건축년도', '법정동명']).copy()
data_cleaned = data_cleaned[data_cleaned['건축년도'] >= 1900].copy() # 건축년도 필터링 (1900년 이상)

# 파생 변수 생성
data_cleaned['연식'] = 2024 - data_cleaned['건축년도']  # 건축 연도 기준 현재 연도까지의 경과 연수

# 로그 변환 적용 (종속 변수)
data_cleaned['물건금액_log'] = np.log1p(data_cleaned['물건금액(만원)'])

# 학습 데이터에서 법정동명 제거
data_for_model = data_cleaned.drop(columns=['법정동명'])

# 데이터 확인
print('데이터셋 크기 : ', data_for_model.shape)
data_for_model.head()
```

```
데이터셋 크기 :  (17306, 6)
```

	물건금액(만원)	건물면적(㎡)	층	건축년도	연식	물건금액_log
9	23800	59.160	1	1998	26	10.077483
10	9700	30.030	8	2001	23	9.179984
11	21000	43.315	3	2012	12	9.952325
12	21000	43.155	4	2012	12	9.952325
21	21500	43.155	3	2012	12	9.975855

데이터 전처리 과정으로 위의 데이터셋(csv파일)에서 회귀분석에 필요한 열만 선택하여 변수에 저장한다.[변수명=data_selected] 또 해당 데이터셋은 서울시의 거의 모든 구역의 부동산 거래 정보가 담겨있어 전체 데이터셋 크기만 해도 12만개의 데이터가 담겨있다. 이를 줄이기 위해 지역명[법정동명]이 '오류동', '온수동'인 지역의 데이터만 가져오기 위해 필터링을 해준다.

오차를 줄이기 위해 개선된 부분.

1. **파생 변수 생성** : 알려지지 않고, 수집되지 않거나 잘못 입력된 데이터 셋의 값들중 결측값들을 처리해주고, 건축년도를 이용해 현재 년도 기준 몇 년이 지난 건물인지 계산하는 "연식" 변수 추가. 위해 파생변수를 생성함.
2. **로그 변환** : 종속변수인 건물금액(컬럼명은 물건금액)의 분포가 너무 큰 경우, 측정이 불가능한 inf값이 나오게 되므로, 정규에 가깝게 로그 변환하여 만들어 모델의 안전성을 높임.[넘파이의 np.log1p 사용]

원래는 건물의 특징으로 범주형 데이터인 "법정동명", "건물 용도" 컬럼의 데이터 값들도 불러와야 하지만, 범주형 데이터이기에 one-hot encoding 기법을 추가해 처리해야하고, 코드가 더 복잡해지고 어려워지기에 두 개의 데이터는 생략하고, 건물의 층 수, 건축 연식, 건물 면적의 데이터만으로 학습시키기로 하였다.
마지막으로 데이터 필터링을 위해 사용했던 범주형 데이터인 "법정동명"은 학습시킬 데이터에서 제외시키고[drop함수 사용] 데이터의 변수명은 data_for_model로 저장한다.
최종적으로 데이터를 확인하면 전체 데이터셋 12만개의 데이터셋중 17306개를 필터링하여 데이터 전처리 작업을 완료하였다.

★3. 모델 객체를 생성하고 훈련데이터를 이용해 학습

```python
from sklearn.model_selection import train_test_split, RandomizedSearchCV, cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error, r2_score

# 독립 변수와 종속 변수 설정
X = data_for_model.drop(['물건금액(만원)', '물건금액_log'], axis=1)
y = data_for_model['물건금액_log']  # 로그 변환된 종속 변수

# 데이터 스케일링
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 데이터셋 분리 (train_size=0.7로 훈련 데이터 크기를 줄임)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
                        test_size=0.3, train_size=0.7, random_state=42)

# 랜덤 포레스트 하이퍼파라미터 최적화
param_dist = {
    'n_estimators': [100, 200],  # 트리 수 줄임
    'max_depth': [10, 15],  # 최대 깊이
    'min_samples_split': [2, 5],  # 분할에 필요한 최소 샘플 수
    'min_samples_leaf': [1, 2],  # 리프에 필요한 최소 샘플 수
    'max_features': ['sqrt']  # 더 작은 범위 탐색
}
```

```python
# 랜덤 포레스트 모델 생성
rf = RandomForestRegressor(random_state=42)

# RandomizedSearchCV 설정 (n_iter=10으로 탐색 시간 단축)
random_search = RandomizedSearchCV(rf, param_distributions=param_dist, n_iter=10, cv=3,
                                   scoring='neg_mean_squared_error', random_state=42, n_jobs=-1)
random_search.fit(X_train, y_train)

# 최적 모델로 학습
best_model = random_search.best_estimator_
best_model.fit(X_train, y_train)

# 데이터 원본 값 복원 (역변환)
X_test_orig = scaler.inverse_transform(X_test)  # 역변환
X_test_orig_df = pd.DataFrame(X_test_orig, columns=X.columns)  # 원본 피처 이름 매핑

# 예측 및 평가
y_pred_log = best_model.predict(X_test)
y_pred = np.expm1(y_pred_log)  # 로그 변환 복원
y_test_orig = np.expm1(y_test)  # 로그 변환 복원
```

```python
# 평가 지표 계산
mse = mean_squared_error(y_test_orig, y_pred)
mape = mean_absolute_percentage_error(y_test_orig, y_pred) * 100
r2 = r2_score(y_test_orig, y_pred)

# 결과 출력
print(f'Mean squared Error: {mse:.2f}')
print(f'R^2 Score: {r2:.2f}')
print(f'MAPE (오차 비율): {mape:.2f}%')
```

```
/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820: Runtim
  _data = np.array(data, dtype=dtype, copy=copy,
Mean squared Error: 61315093.66
R^2 Score: 0.68
MAPE (오차 비율): 18.53%
```

파이썬에서 사용하는 머신러닝 라이브러리중 하나인 사이킷런(scikit- learn)을 임포트하여 여러 머신러닝 알고리즘과 여러 모듈들을 이용해 회귀(Regression)분석을 진행한다. 하나하나 살펴보면

① 데이터를 학습용과 테스트용으로 나눌 때 사용하는 train_test_split

② 머신러닝 알고리즘이 예측한 값과 실제값의 차이를 의미하는 오차를 제곱한 값의 평균인 "평균제곱오차"(MSE:mean_squared_error)

	③ 회귀 모델이 잘 학습되었는지 확인하는 평가지표, 오차가 어느정도인지를 백분율값으로 나타내는 MAPE(mean_absolute_percentage_error) ④ 결정계수 R^2를 계산하여 모델 성능을 평가하는 r2_score **오차를 줄이기 위해 개선된 부분.** ⑤ 데이터를 표준화(Scaling)하기 위한 클래스 StandardScaler ⑥ 하이퍼파라미터를 랜덤으로 탐색하여 최적의 모델을 찾는 RandomizedSearchCV ⑦ 랜덤으로 생성된 무수히 많은 트리를 이용하여 예측을 하는 랜덤 포레스트 회귀 모델인 RandomForestRegressor 등을 이용해 라이브러리를 가져오고, **(6~8번행)** 독립변수(X)와 종속변수(Y) 데이터를 지정한다. 이때 데이터 전처리 작업때 최종적으로 나온 데이터인 data_ for_model에서 '물건금액(만원)'과 그걸 log값으로 변환했던 '물건금액_log'를 종속변수(Y)로 지정, 그리고 이들을 drop한 나머지 컬럼들을 독립변수(X)로 지정한다. **(10~12번행)** StandScaler 클래스를 이용해 독립변수(X) 각각의 데이터를 평균 0, 분산 1로 조정하여 각 특징들 간의 값 범위 차이를 줄이고, **(14~16번행)** train_test_split을 이용해 데이터셋을 훈련용(70%)와 테스트용(30%)로 분리함. **(19~33번행)** 분류와 회귀분석 등에 사용되는 **앙상블 학습 방법**(좋은 예측 성능을 얻기위해 다수의 학습 알고리즘을 사용하는 방법) 중 일종인, 랜덤 포레스트의 하이퍼파라미터 튜닝(최적화)를 진행한다. param_dist는 하이퍼파라미터의 탐색 공간을 정의하기 위해 다음의 매개변수들을 이용한다. ◆ n_estimators: 랜덤 포레스트의 트리 개수. ◆ max_depth: 트리의 최대 깊이. ◆ min_samples_split: 분할에 필요한 최소 샘플 수. ◆ min_samples_leaf: 리프 노드에 필요한 최소 샘플 수. ◆ max_features: 각 노드에서 사용할 최대 특성 수. 이렇게 하이퍼 파라미터를 어떻게 설정하느냐에 따라 모델의 성능이 달라지며, 최적의 하이퍼파라미터를 찾기 위해 흔히 사용하는 방법으로 *GridSearchCV()와 RandomizedSearchCV()* 가 있다. 위에서 지정한 매개변수들을 가지고 모든 조합에 대해 교차검증 후 가장 좋은 성능을 내는 하이퍼파라미터의 조합을 찾는 것이 *GridSearchCV()* 모듈이다. 이와 동일한 방식으로 사용되나, 모든 조합을 시도하지는 않고, 지정한 반복마다 임의의 값만 대입해 지정한 횟수만큼 평가하는 것이 *RandomizedSearchCV()* 모듈이다. 여기서는 후자를 선택해 모델을 탐색할 것이다. 그래서 param_dist의 설정값들을 가지고 랜덤으로 10번(n_iter=10) 파라미터를 조합해 최적의 모델을 탐색하고, 교차검증(cv=3)을 통해 검증 데이터로 평가한 것을 random_search 변수에 저장하여 fit()함수를 통해 최적의 하이퍼파라미터를 찾아낸다. **(35~37번행)** 앞서 진행한 최적의 하이퍼파라미터를 가지고 랜덤포레스트 모델인 best_model[최적의 모델]을 생성한다. 그리고 훈련데이터를 이용해 모델을 학습(fit())시킨다. **(39~41번행)** 테스트 데이터(X_test)를 스케일링 이전 값으로 복원시킨뒤, 이후에 시각화 플롯을 위해 원래의 특징 이름(X.columns)을 사용해 데이터프레임(df)를 생성함. **(43~46번행)** 로그로 변환된 값을 예측(predict)하고, 모델이 예측한 로그 변환값인(y_pred_log)를 복원시키고 실제값인(y_test)도 로그를 복원시켜 동일

한 스케일로 변환한다.
(48~56번행) 이제 모델이 잘 학습되었는지 확인하기위한 평가 지표를 계산하는 부분으로 실제값과 예측값의 오차 제곱 평균(MSE)와 예측값과 실제값 간의 평균 백분율 오차(MAPE)를 나타낸다. 이 두 값 모두 낮을수록 오차가 적고, 정확도가 높다는 뜻이다. 추가로 예측값이 실제값의 변동성을 얼마나 설명하는지 나타내는 지표인 R2도 있으며 1에 가까울수록 모델 성능이 좋다.

```
14 # 선형 회귀 모델 학습
15 model = LinearRegression()
16 model.fit(X_train, y_train)
17
18 # 예측값 계산
19 y_pred = model.predict(X_test)
20
21 # 성능 평가 (MAE와 R^2)
22 mae = mean_squared_error(y_test, y_pred)
23 r2 = r2_score(y_test, y_pred)
24 mape = mean_absolute_percentage_error(y_test, y_pred) * 100
25
26 print(f'Mean Absolute Error: {mae}')
27 print(f'R^2 Score: {r2}')
28 print(f"MAPE: {mape:.2f}%")
```

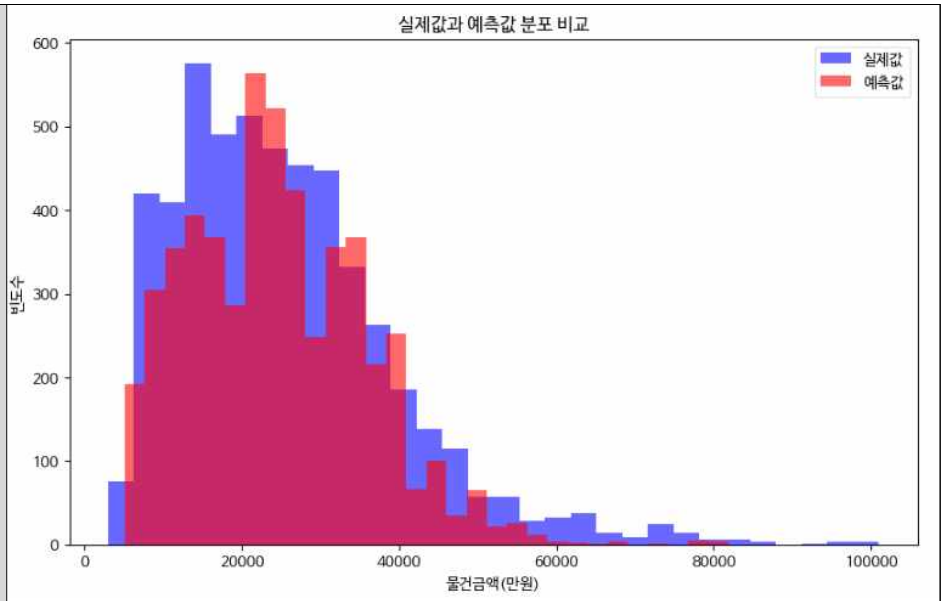
```
Mean Absolute Error: 128899884.96319562
R^2 Score: 0.4803887303134947
MAPE: 34.46%
```

초기 구성 모델[LinearRegression 선형 회귀 알고리즘]는 오차가 너무 컸다.
아주 기초적인 알고리즘이기에 최신 알고리즘에 비해선 예측력이 떨어지곤 했다.

★4. 회귀분석 결과를 시각화하기

```
1 import matplotlib.pyplot as plt
2 plt.rc('font', family='NanumBarunGothic')
3 # 실제값과 예측값 히스토그램
4 plt.figure(figsize=(10, 6))
5 plt.hist(y_test_orig, bins=30, color='blue', alpha=0.6, label='실제값')
6 plt.hist(y_pred, bins=30, color='red', alpha=0.6, label='예측값')
7 plt.xlabel("물건금액(만원)")
8 plt.ylabel("빈도수")
9 plt.legend()
10 plt.title("실제값과 예측값 분포 비교")
11 plt.show()
```

(1번행) 시각화에 필요한 모듈을 불러옴
(4번행) figsize를 이용해 그래프의 크기를 설정
(5~6번행) 실제값의 히스토그램과 예측값의 히스토그램을 생성함. 실제값은 y_test_orig 변수를 이용해 테스트 데이터의 실제값을 가져오고 히스토그램의 구간을 30개로 나눔(bins=30), 히스토그램 색상은 파란색(color='blue')으로 하고, 히스토그램의 투명도를 0.6(alpha=0.6)으로 하여 예측값과 겹칠 때 두 분포를 동시에 확인한다. 범례(label)에 표시할 이름은 '실제값'으로 지정. 예측값의 히스토그램도 위와 동일하다. 단지 y_pred 변수를 이용하고 색상을 바꾸는 것 뿐.
(7~11번행) 이후 x축, y축 라벨을 설정하고 plt.legend()를 이용해 히스토그램에 표시하고 plt.show()로 작성된 히스토그램을 화면에 표시한다.



히스토그램의 겹침이 비슷하게 겹칠수록 모델이 실제값을 잘 예측하고 있단 것이다. 각 구간에서 두 값의 빈도 차이를 확인해 모델이 특정 금액대에서 잘 맞는지 아니면 벗어난 경향이 있는지 판단이 가능함. 하지만 현재 데이터 셋의 값이 여러 개선사항을 적용했음에도 분포 차이가 큰 관계로 어느정도 차이가 나고 있음을 알 수 있다. 이럴 수밖에 없던 이유는 나중에 프로젝트를 하며 느낀점에서 후술할 예정.

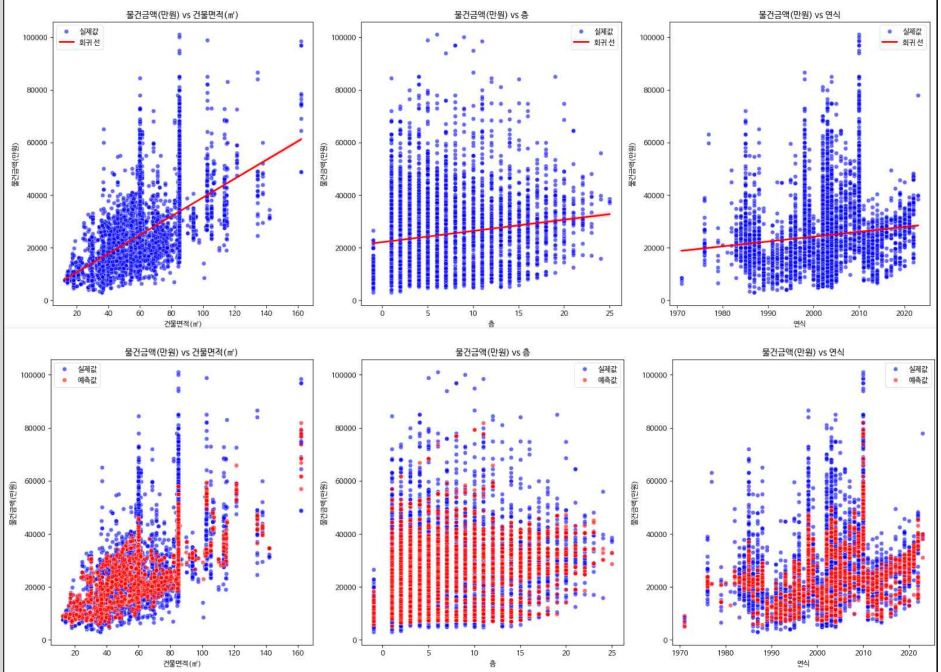
아무튼 이렇게 히스토그램으로 얻을수 있는 것은 모델 성능의 직관적인 평가가 가능하고, 데이터 분포를 통해 모델이 어느 구간에서 성능이 낮은지 확인이 가능하다.

```python
import matplotlib.pyplot as plt
import seaborn as sns
plt.rc('font', family='NanumBarunGothic')

# 예측된 값과 실제 값의 산점도를 각 특성에 대해 그리기
features = ['건물면적(㎡)', '층', '연식']  # x_feature

plt.figure(figsize=(18, 12))

# 각 피처별로 그래프 그리기
for i, feature in enumerate(features):
    plt.subplot(2, 3, i+1)  # 2행 3열의 서브플롯

    # 산점도 그리기
    sns.scatterplot(x=X_test_orig[:, i], y=y_test_orig, color='blue', label='실제값', alpha=0.6)
    sns.scatterplot(x=X_test_orig[:, i], y=y_pred, color='red', label='예측값', alpha=0.6)

    # 선형 회귀선 그리기
    sns.regplot(x=X_test_orig [:, i], y=y_pred, scatter=False, color='green', label='회귀 선')

    plt.title(f'물건금액(만원) vs {feature}')
    plt.xlabel(feature)
    plt.ylabel('물건금액(만원)')
    plt.legend()

plt.tight_layout()  # 레이아웃 정리
plt.show()
```

앞에선 히스토그램을 이용해 실제값과 예측값의 분포를 통해 모델성능을 확인하였고, 이번에는 모델성능을 각 특징(건물면적, 층, 연식)별로 실제값과 예측값의 관계를 시각적으로 확인한다.

(6번행) 그래프의 x축이 될 특성들을 features 변수에 저장한다.

(11~19번행) 각 피처별로 그래프를 그리기 위해 반복문을 이용해 서브플롯을 생성한다. #특징이 3가지 이므로 3가지의 특징에 대한 시각화를 진행해야 하기에 plt.subplot을 사용한다. 6행에서 선정한 features의 인덱스(i)와 이름을 통해 반복적으로 처리한다.[enumerate(features)]
plt.subplot(2, 3, i+1) : 2행 3열의 서브플롯에서 현재 위치에 그래프를 추가함. 첫 번째 그래프의 위치는 (1,1), 두 번째는 (1,2)..순으로 추가됨
데이터의 분포를 직관적으로 보기위해 산점도를 그리는 함수 sns.scatterplot()를 이용하는데 X축 데이터로 사용될 특징들을 가져오며 최적화를 위해 사용했던 데이터 스케일링 이전의 원본 데이터 X_test_orig를 사용함. 그리고 Y축 데이터로 실제값(y_test_orig), 예측값(y_pred)을 사용하고, 색상과 label값 지정, 점의 투명도를 지정한다.
그리고 데이터에 대한 모델의 예측 추세를 나타낼 땐 회귀선만 이용하게 되며 sns.regplot()을 이용한다. scatter=False로 산점도를 제외시키고, X축의 특성데이터, Y축의 예측값을 사용함.



★5. 건물의 특징에 임의의 데이터 값을 넣고 건물가격을 예측함.

```
print("건물 가격을 예측하고 싶은 건물의 정보를 입력해주세요.")

floor = int(input("층수 : "))
area_size = int(input("건물 면적(㎡) : "))
year = int(input("건축년도 : "))
```

```
건물 가격을 예측하고 싶은 건물의 정보를 입력해주세요.
층수 : 3
건물 면적(㎡) : 70
건축년도 : 2018
```

```
user_input = [[area_size, floor, year, 0]]  # '연수'는 사용하지 않을것이므로 0으로 채움 (훈련 데이터에 맞춤)
# 스케일링 후 예측
user_input_scaled = scaler.transform(user_input)
building_predict_log = best_model.predict(user_input_scaled)  # 예측 (로그 변환된 값)
building_predict_price = np.expm1(building_predict_log)  # 로그 변환 복원
```

숨겨진 출력 표시

```
print(f"이 건물의 예상 가격은 {building_predict_price[0]:,.2f}만원 입니다.")
```

```
이 건물의 예상 가격은 34,060.39만원 입니다.
```

	예측하는데에 이용했던 층수, 건물면적, 건축년도를 입력받아서 user_input 변수에 저장하고, 학습시키는데 사용했던 모델을 사용하기 위해 스케일링 작업[scaler.transform()]후 예측을 진행한다. 그리고 스케일링작업을 했던 로그 변환된 값을 복원시키고 예측 결과를 출력하는 과정이다.
기대 효과 및 활용방안	기대 효과로는 부동산에 의존할 필요가 없고, 부동산 데이터만 존재한다면 추가적인 조사 없이 빠르게 가격 예측이 가능하다는 점과 과거의 데이터를 기반으로 한 예측이기에 주관적인 판단보다 신뢰도가 높다는 점과 건물의 특징(면적, 층수, 건축년도 등)의 변수가 건물 가격에 미치는 영향을 시각적으로 분석해, 가격 형성 요인에 대한 이해가 쉽다는 점이 있다. 활용 방안으로는 부동산 투자 결정이나 부동산 중개업과 같이 거래부문에서도 활용될 것이고, 지역별 부동산 정책 및 개발에 적정 가격을 분석하는 자료로 이용되거나 부동산과 관련된 애플리케이션을 만드는데에도 활용될 수 있을 것이다.
2024 년 11 월 21 일 **지도교수 금 득 규 (인)**	