

Members

김승호, 교육학과, koridore987@hanyang.ac.kr

신지호, 기계공학부, sjho4311@hanyang.ac.kr

박소미, 미디어커뮤니케이션학과, zzyan_9@naver.com

중점 활동

팀원명	내용
김승호	데이터셋 탐색, 지오크딩, 데이터셋 피쳐엔지니어링, 데이터분석 코딩
신지호	데이터셋 탐색, 데이터셋 시각화
박소미	서론 작성 보조, 블로그 내용 작성 보조

I. 서론

한국에서의 사교육은 큰 중요성을 지닌다. 한국에서 사교육은 가계의 지출 1순위이자 사회 문제 중 하나로 꼽힌다. 학벌주의 사회 내에서 사교육은 계층 상승을 위한 수단으로 여겨지며, 부모는 자녀의 성공을 위해 사교육비 지출에 큰 힘을 쏟는다. 이러한 경향은 꾸준히 강화되는 추세를 보이고 있다.

2023년 7월 교육부와 통계청이 조사한 '2022년 초등학교 사교육비 조사' 결과, 사교육 참여율과 사교육비가 역대 최대인 것으로 나타났다. 사교육비는 총 26조로 2021년에 달성한 최대치에 10.8%나 더해진 수치이다. 사교육 참여율은 78.3%로 10명 중 8명 가까이 사교육을 듣는다고 할 수 있다. 심지어 초등생의 사교육 참여율은 85.2%로 10명 중 8명 이상이 사교육을 받고 있는 것으로 나타났다. 매년 증가하는 사교육을 통해 알 수 있듯이 사교육은 이미 한국의 교육사회에 깊숙히 자리잡았다.¹

사교육 시장은 점차 늘어나고 있지만, 실제 수요자라 볼릴 수 있는 학생의 수는 줄어들고 있다는 점이 주목할만 하다. 총량은 늘어나고 있지만, 서비스를 받는 고객이 줄어든다는 사실은 인당 부담해야 하는 비용이 증가하고 있다는 결론에 도달한다. 이는 점차 부담할 수 있는 가정과 그렇지 않은 가정으로 분리되는 상황으로 전개되며 경제적 격차가 교육의 격차를 만들 수 있다는 점을 시사한다.

사교육비가 늘어나게 된 요인은 무엇인가. 그리고 사교육비를 책정하는 요인들에는 어떤 것이 있는가. 다음과 같은 질문에 답하기 위해 본 팀프로젝트에서는 사교육의 중심이라 할 수 있는 서울 지역의 사교육비의 실태를 조사하고자 한다. 구체적으로는 다음과 같다.(미결)

1. 교습소의 특징(학원 과목, 수강생, 교습과정, 위치)등과 교습소를 둘러싼 환경적 변수(주변 학교의 특징, 교통편의 형태 등)을 통해 현재 사교육비에 주요하게 미치는 요인이 무엇인지 살펴본다.
2. 특정한 조건을 가진 교습소가 어느정도의 교습비를 책정해야 하는지 예측하고, 현재 고평가된 (혹은 저평가된) 교습소는 어느 곳인지 살펴본다

학원에는 단과학원, 종합학원, 재수학원, 편입학원 등 입시 전문 학원과 공무원 학원 등을 포함하는 취업전문학원, 그리고 미술 학원, 음악학원 등 예체능 전문학원 같이 다양한 종류가 있지만 본 프로젝트에서는 입시 전문 학원 중 (국어, 영어, 수학 등 정해진 과목을 가르치는) 단과학원과 종합학원에 한정하여 모델을 만들고자 한다.

본 프로젝트에서는 서울 내 지역별 사교육비 실태를 분석하고, 나아가 특성별 사교육비를 예측하는 것을 목표로 한다. 또한 어떤 모형이 사교육비 예측에 적합한지 평가하고 최적의 모형을 선택, 분석할 것이다.

II. 데이터 설명

사용한 데이터셋은 총 2가지로, 세부 데이터셋의 설명은 다음과 같다.

1. 서울특별시교육청 학원 및 교습소 등록(신고) 교습비 현황

전국의 교습소는 「학원의 설립·운영 및 과외교습에 관한 법률」 제15조의5(정보의 공개)에 따라 학원 등록 및 교습소 신고 교습비등 현황 공개해야 한다. 본 팀은 관련 법률에 의거해 공공데이터포털에 업로드된 데이터를 사용하였다. 업로드된 데이터는 서울시의 교육지원청별로 구분되어 작성되었다. 데이터는 서울에 있는 모든 11개의 지원청의 내용을 종합해 작성하였다. 데이터에 대한 자세한 정보는 다음과 같다.

구분	내용	비고
제공기관	서울특별시교육청	
관리부서명	평생교육과	
등록일	2020-05-29	
표본수	238,088	
변수목록	관할교육청, 학원명, 학원종류, 분야구분, 학원주소, 교습과정, 교습과목(반) 등	총 23개의 열

2. 서울시 학교 기본정보

이번 분석의 파생변수에 학원 주변의 학교 정보를 추가하기로 결정했다. 학교와 관련된(혹은 초·중·고등학교 학생과 관련된) 변수가 교습비에 영향을 미칠 것이라 생각했기 때문이다. 학원의 주변에 있는 학교를 찾기 위해 우선 학교 위치 데이터를 사용했다. 위치 정보는 공공데이터포털에 업로드된 '전국초·중·고등학교위치표준데이터'를 사용하였다. 데이터의 자세한 정보는 아래와 같다.

구분	내용	비고
소관기관	교육부	
제공기관	청주대학교 지방교육재정연구원	
표준데이터셋제공시스템	학구도안내서비스	
등록일	2023-04-18	11,989
표본 개수		
변수목록	학교ID, 학교명, 설립형태, 교육지원청명, 위도, 경도 등	총 18개의 열

또한 관련 학교의 정보를 얻기 위해 추가적인 학교 현황 데이터를 사용하였다. 학교 정보는 학교 알리미에 공개되어있는 자료를 수집, 가공하였다.

```
df_aca = pd.read_csv("{file_path}", index_col=0) # 교습비 데이터
df_sch = pd.read_csv("{file_path}", index_col=0) # 학교 현황 데이터
```

III. 데이터 전처리

- Explaining your choice of algorithms (methods)
- Explaining features (if any)

1. 결측치 처리

학교 데이터를 받아오는 과정에서 학교 현황데이터와 학교 위치 데이터의 수집 시점이 일치하지 않아서 두 데이터를 병합하는 과정에서 발생하는 결측치가 존재했다. 학교 정보가 변경되었거나, 현재 폐교 등의 행정조치로 사라진 학교 등이다. 사라지거나 변경, 이전된 학교는 총 7개, 데이터가 변한 학교는 총 1개였다. 각 레이블을 확인해 직접 제거하는 코드를 작성, 실행했다.

구분 대상 학교

변경 상일중학교(기존 상일여자중학교에서 공학으로 전환)

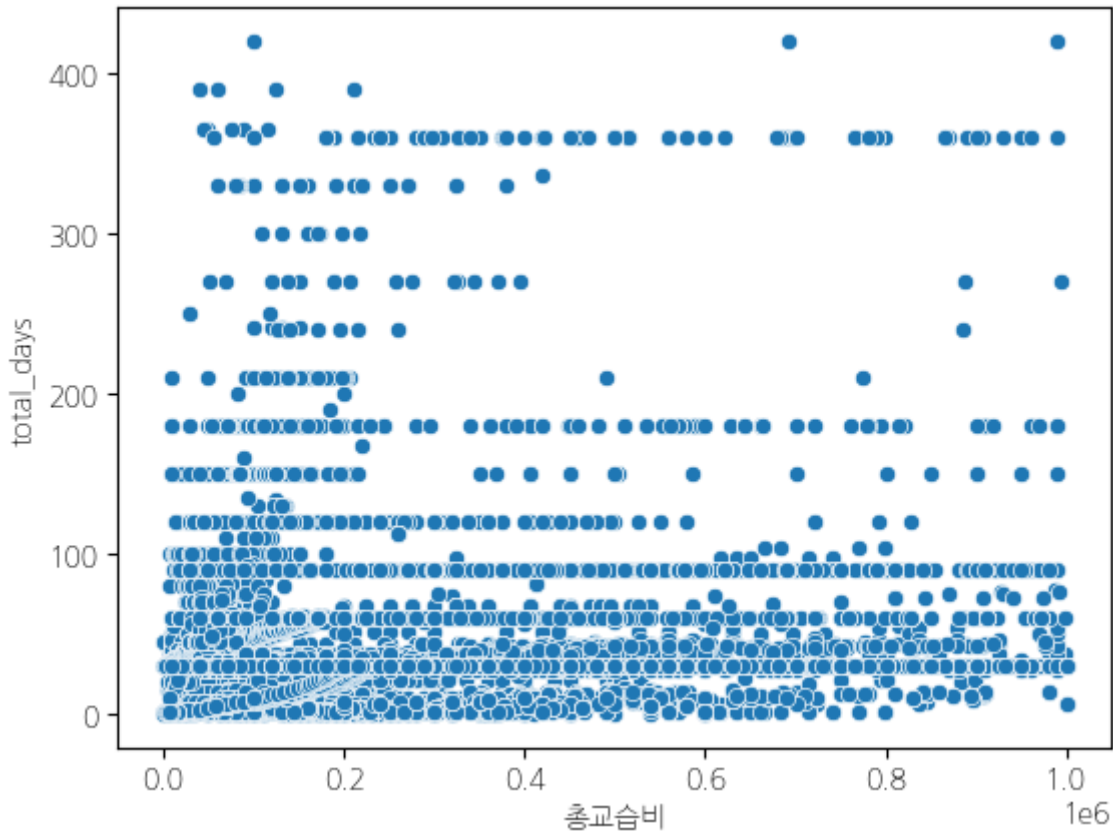
삭제 덕수고등학교, 개포중학교, 서울화양초등학교, 서울개원초등학교, 서울둔촌초등학교, 서울위례초등학교

또한 학원 데이터의 결측치도 존재했다. 그 중 데이터 분석에 유의미한 영향을 미치는 결측치는 총 34개 학원으로, 교습계열, 교습시간, 총교습비 등 데이터 분석에 필요한 열의 값을 가지고 있지 않았다. 따라서 이 데이터들도 모두 삭제 처리 했다.

```
df_aca.dropna(subset=['관할교육청', '학원명', '학원종류', '분야구분', '학원주소', '교습계열',
                     '교습과정', '교습과목(반)', '정원', '교습기간', '총교습시간(분)', '교습비', '모의고사비', '재료비',
                     '급식비', '기숙사비', '차량비', '피복비', '기타경비합계', '총교습비', '강사수', '위도', '경도'], inplace = True)
df_aca['months'] = df_aca.dropna()['교습기간'].str.extract(r'(\d+)개월').astype(int)
df_aca['days'] = df_aca.dropna()['교습기간'].str.extract(r'(\d+)일').astype(int)
df_aca['total_days'] = df_aca['months'] * 30 + df_aca['days']
```

2. 이상치 처리

교습소 교습비에 이상치가 존재했고, 이를 처리하는 과정을 거쳤다. 이상치가 발생한 이유로는 학원별로 교습과정을 설정하고 산출하는 방법이 다양하기 때문인 것으로 파악했다. 일례로 학원 교습비의 할인 내용을 교습과정에 포함시키기도 하고, 무료체험 등의 과정을 넣기도 해 교습비가 0인 값들이 존재했다. 또한 단위 역시 다양했는데, 원 단위가 아닌 천원 단위이거나 만원 단위인 것으로 의심되는 값들도 존재했다. 학원 교습과정의 교습비가 한 두 자리수인 경우 이에 해당한다고 보았다.



따라서 본 프로젝트에서는 팀원들의 일관된 의견에 따라 다음 기준으로 데이터를 걸러내었다.

1. 교습 기간이 한달(1개월 0일)로 작성되어 있는 행
2. 학원 종류가 평생직업교육학원이 아닌 행
3. 모의고사비, 재료비, 급식비 등 기타 교습비용이 0인 행
4. 총교습비가 5만원 이상 100만원 이하인 행

데이터에 대한 처리 코드는 다음과 같다.

```
# 평생직업교육학원이 아닌 컬럼만 추출
df_aca = df_aca[df_aca.학원종류 != "평생직업교육학원"]

# 기타 교습비용 0인 행 제거
drop_list = ['모의고사비', '재료비', '급식비', '기숙사비', '차량비', '피복비', '기타경비', '합계']
for i in drop_list:
    df_aca = df_aca[df_aca[i] == 0]
df_aca = df_aca.drop(columns=drop_list)

# 총교습비가 5만원 이상 100만원 이하인 행 추출
df_aca = df_aca[df_aca.총교습비 >= 50000][df_aca.총교습비 <= 1000000]

# 교습기간이 한달로 작성되지 않은 행 제거
df_aca = df_aca[df_aca.months == 1][df_aca.days == 0]
```

3. 지오크딩을 이용한 학교 데이터 편집

3-(1). 네이버 지도 API를 이용한 지오코딩

이번 분석에서, 학원에서 일정 거리에 떨어져있는 학교의 특성을 반영한 데이터를 작성하고자 했다. 하지만 학원 데이터에는 주소만 나와있어서 거리를 측정해 새로운 변수를 만들어 내기는 어려운 상황이었다. 따라서 기존에 가지고 있는 학원의 주소 데이터를 가지고 위도와 경도 데이터로 변환해 거리를 측정하기로 하였다. 지오코딩은 네이버 지도의 API²를 이용해 주소에 대응하는 위도와 경도를 입력했다. geocoding 과정에서 발생하는 결측치는 삭제처리했다. 학원이 없어졌거나, 지도 상에 등록되어 있지 않은 학원이거나, 학원의 위치가 실제와 다른 경우들이었다.

```
import pandas as pd
from urllib.request import urlopen
from urllib import parse
from urllib.request import Request
from urllib.error import HTTPError
import json

#naver map api key
client_id = '{팀원 아이디}';
client_pw = '{팀원이 할당받은 키}';
api_url = 'https://naveropenapi.apigw.ntruss.com/map-geocode/v2/geocode?query='
```

```
# 네이버 지도 API 이용해서 위경도 찾기
def geocoding(add):
    add_urlenc = parse.quote(add)
    url = api_url + add_urlenc
    request = Request(url)
    request.add_header('X-NCP-APIGW-API-KEY-ID', client_id)
    request.add_header('X-NCP-APIGW-API-KEY', client_pw)
    try:
        response = urlopen(request)
    except HTTPError as e:
        print('HTTP Error!')
        latitude = None
        longitude = None
    else:
        rescode = response.getcode() #정상이면 200 리턴
        if rescode == 200:
            response_body = response.read().decode('utf-8')
            response_body = json.loads(response_body) # json
            if response_body['addresses'] == [] :
                print("'result' not exist!")
                latitude = None
                longitude = None
            else:
                latitude = response_body['addresses'][0]['y']
                longitude = response_body['addresses'][0]['x']
        else:
            print('Response error code : %d' % rescode)
            latitude = None
```

```
longitude = None
return latitude, longitude
```

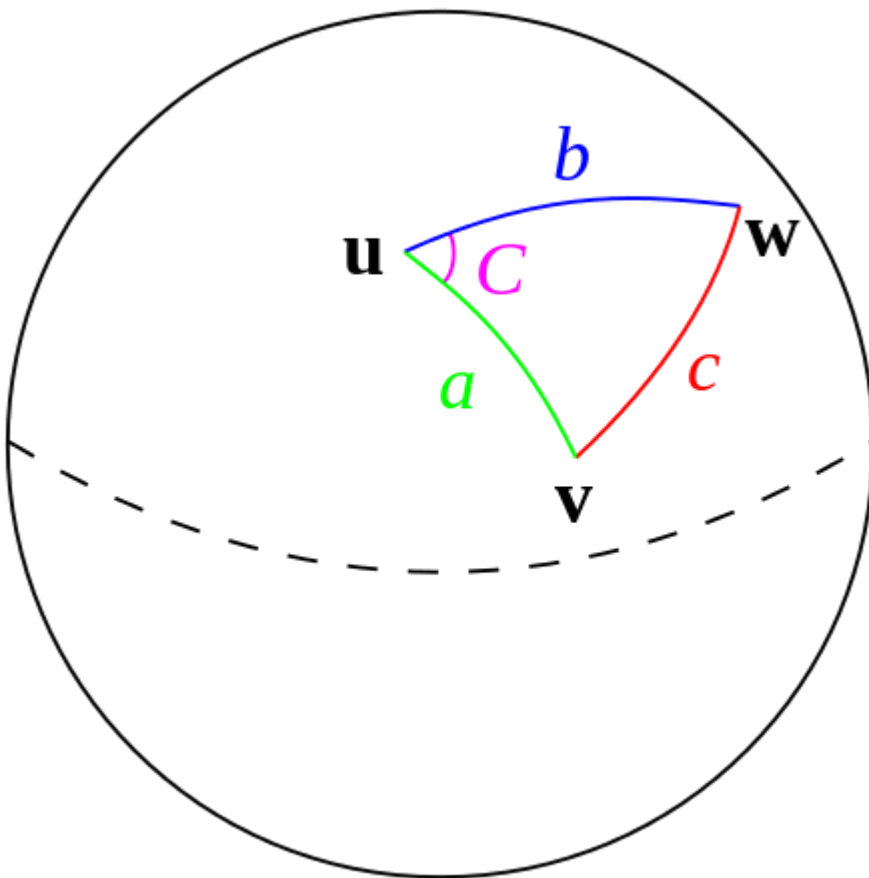
3-(2). 위경도를 통해 근방 1km 이내 학교 구하기

위도와 경도 데이터를 활용해 학원 근방 1km 내에 있는 학교 정보를 가져오기 위해 {BallTree}의 {haversine}구조를 사용했다. haversine 공식은 각 위경도 값을 라디안 값으로 가지기 때문에, {numpy}의 {deg2rad}메서드를 사용해서 각으로 변환 후, 각 지점을 BallTree의 포인트에 대응시켜 거리를 구했다. 단, 이 공식은 지구를 완벽한 구라고 가정하고 곡면의 표면거리를 구하기 때문에 실제와는 오차가 있을 수 있지만, 일반적인 사용에서는 무리가 없다고 판단했다.

```
df_aca_rad = np.deg2rad(df_aca[['위도', '경도']])
df_sch_rad = np.deg2rad(df_sch[['위도', '경도']])

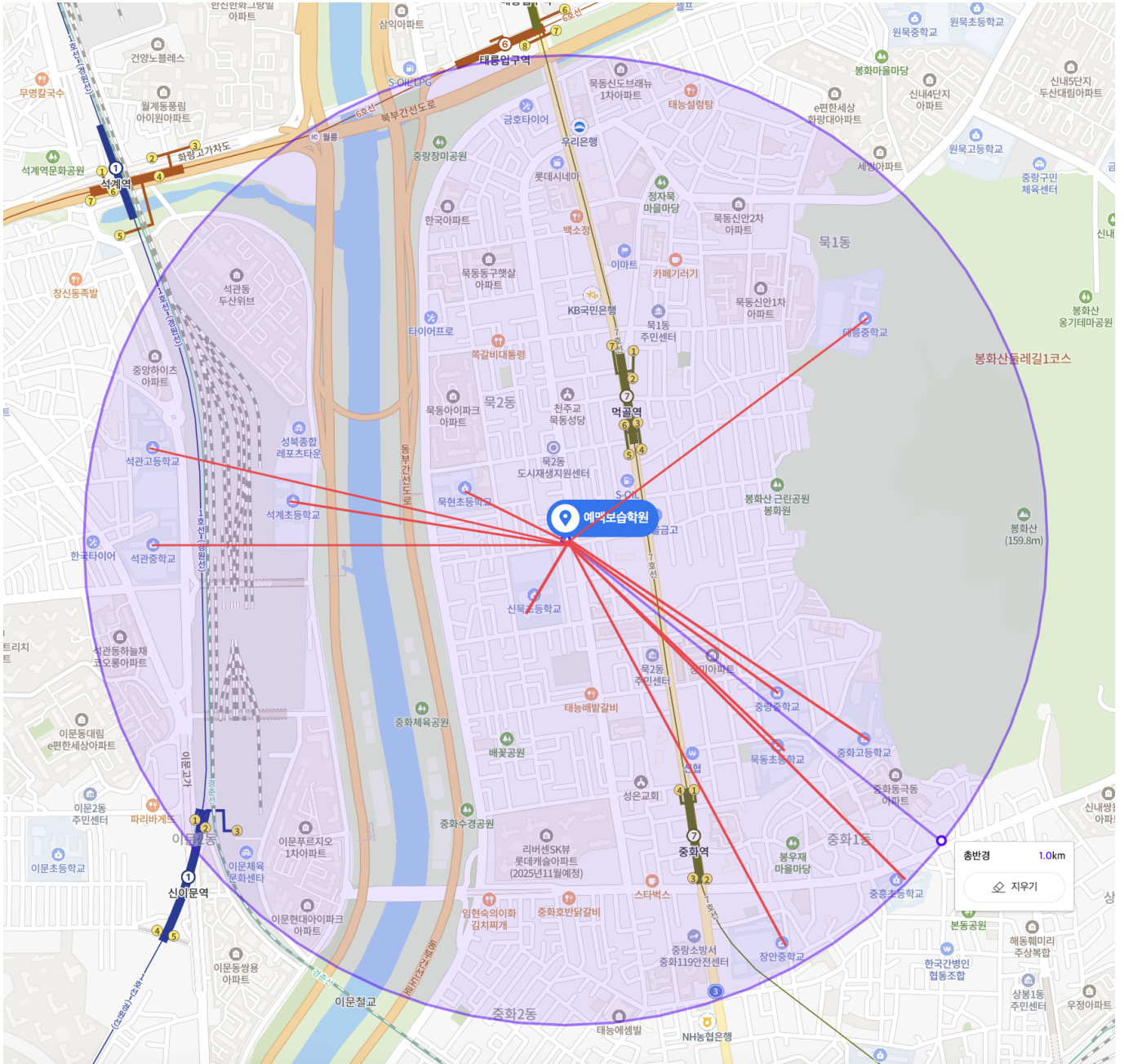
tree = BallTree(df_sch_rad, metric='haversine')
near_idc = []
distcrit = 1 # 1km
for point in df_aca_rad.values:
    idc = tree.query_radius([point], r=distcrit/6371) # 지구 반지름 6371
    near_idc.append(df_sch.loc[idc[0], '학교ID'].tolist())

df_aca['near_idc'] = near_idc
```



$$\text{hav}(c) = \text{hav}(a - b) + \sin(a) \sin(b) \text{hav}(C).$$

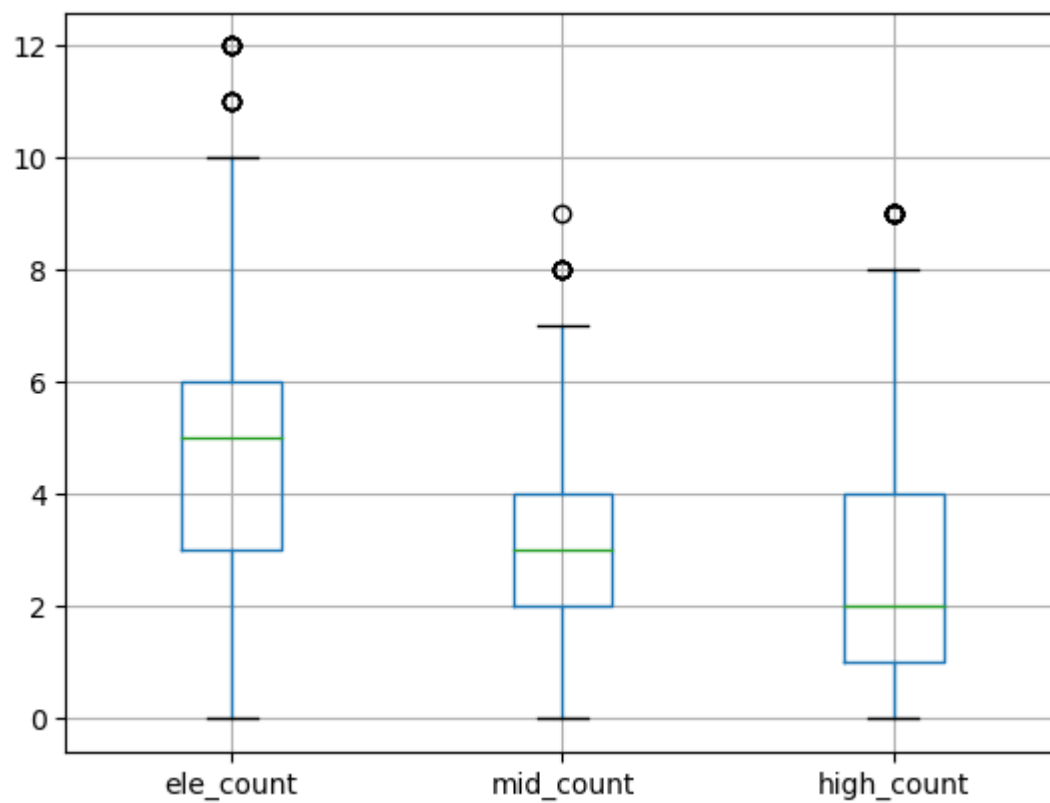
다음은 첫 번째 열에 있던 학원의 근방 학교를 시각화한 것이다.



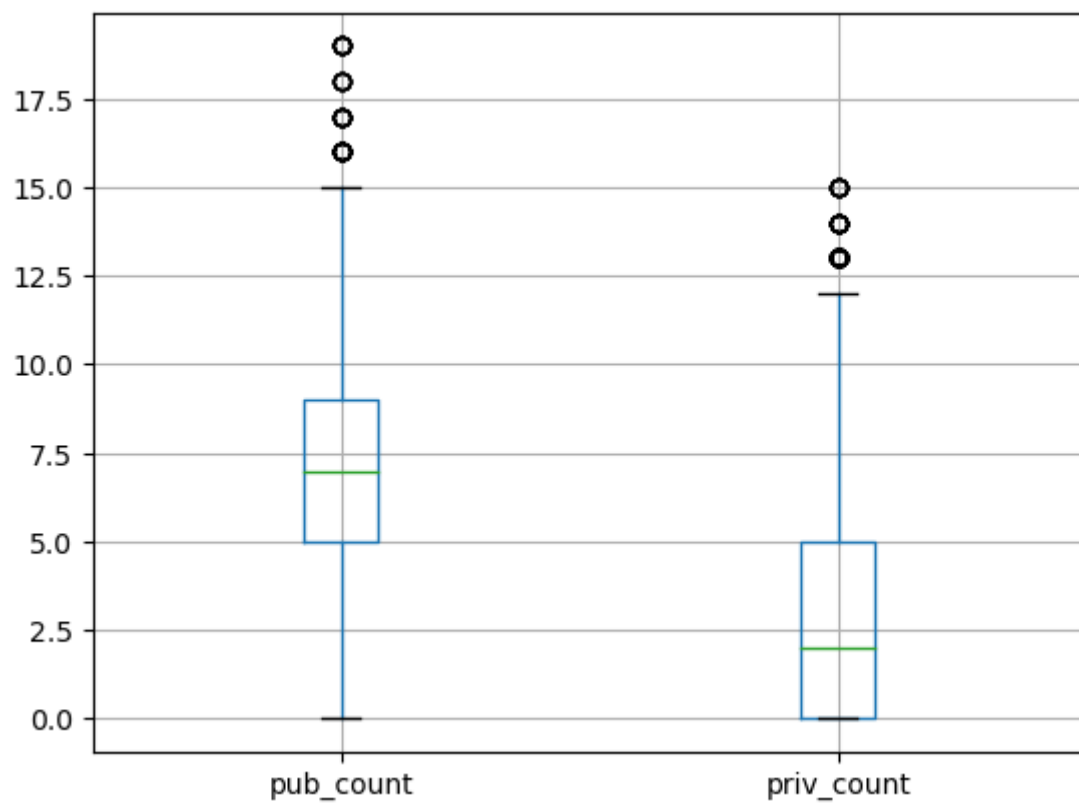
3-(3). 근방 학교 현황 데이터 추출

근방의 학교를 구한 후, 우리는 근방 학교의 학교 급별, 학교 설립유형 별 학교수와 학생수를 새로운 feature로 계산했다. 계산한 feature의 특성은 다음과 같다.

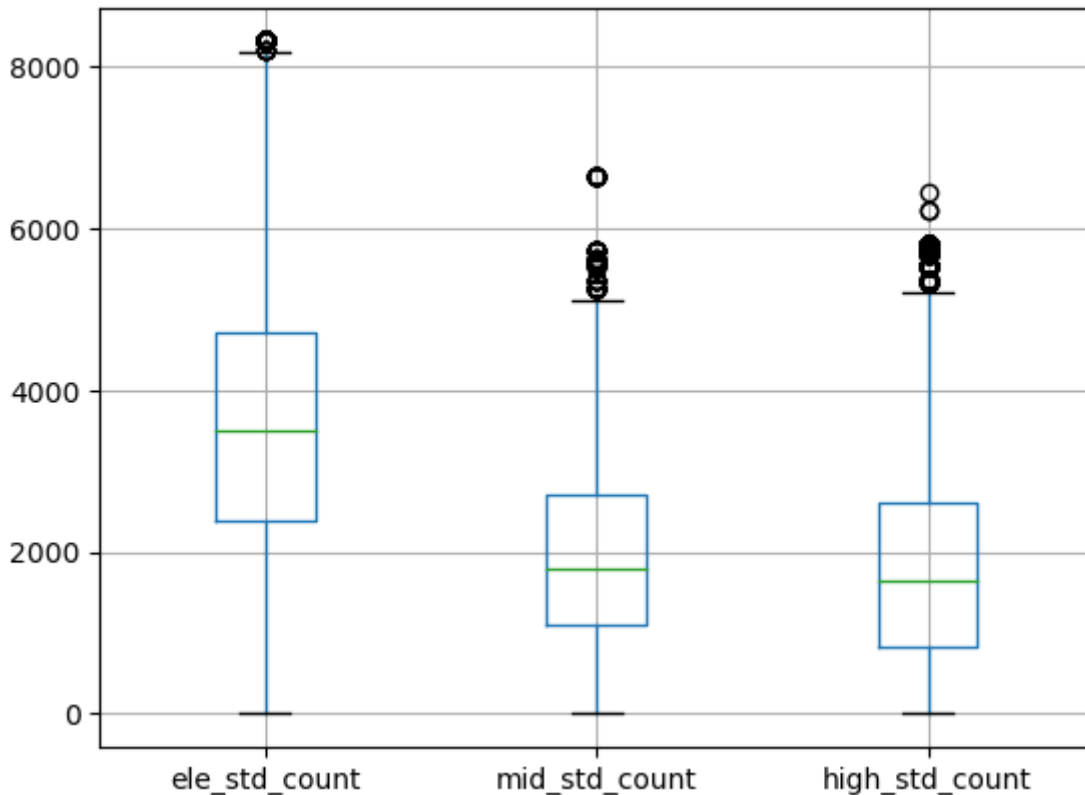
<학교 급별 학생수>



<학교 설립유형별 개수>



<학교 급별 학생수>



자세한 계산 코드는 다음과 같다

```
# 조건에 해당하는 학교 ID 추출해 반환하는 함수
def school_sort_df(target_col, target_value = [], school_ID=[]):
    school_df = df_sch[df_sch['학교ID'].isin(school_ID)]
    result_df = school_df[school_df[target_col].isin(target_value)].copy()
    return result_df

# 학교급별 학교 수와 학생 수 추출
df_group1 = df_aca_loc['near_sch'].groupby(['학교급']).agg({'학교급': 'count',
'학생수(계)': 'sum'}).rename(columns={'학교급': 'count', '학생수(계)':
'std_count'}).reset_index()

# 설립 유형별 학교 수 추출
df_group2 = df_aca_loc['near_sch'].groupby(['설립']).agg({'설립':
'count'}).rename(columns={'설립': 'count'}).reset_index()

df_aca_loc = pd.merge(df_aca_loc, df_group1, on='학교급', how='left')
df_aca_loc = pd.merge(df_aca_loc, df_group2, on='설립', how='left')
```

3. '교습과목(반)'열 데이터 프로세싱

교습과목(반) 열에는 해당 학원에서 진행하는 교육 프로그램에 관한 세부적인 내용이 담겨있다. 하지만 학원별로 과정을 부르는 명칭이 다 다를 뿐만 아니라, 같은 학원에서 진행하는 내용이더라도 그 표현을 다르게 한 경우가 존재했다. 예를 들어 고등학교 2학년 학생을 대상으로 진행하는 국어 내신 대비 교습과정의 경우 한 학원에서는 '고2국어내신'이라고 표현하는 반면, 다른 학원에서는 '국(고등2)'라고 표현하는 등 다양했다. 해당 열은 처리하기 복잡하지만 내용이 많고 교습비를 예측하는 데 가장 큰

기여를 할 것으로 보여 서울대학교에서 연구한 한국어 토큰화 알고리즘 꼬꼬마(Kkma)한국어 형태소 분석기를 활용해 분석하였다.

아래는 토큰화 세부 코드이다.

```
from konlpy.tag import Kkma # 패키지 импорт
kkma = Kkma()

def pos(text):
    tk_list = []
    tk_text = kkma.pos(text)
    tk_list.append(tk_text)
    return tk_list[0]

df['tk_text'] = df['교습과목(반)'].apply(pos)
```

그 후, Kkma 태깅 규칙³에 따라 유효한 태깅만 리스트로 반환해 데이터에 추가하였다.

```
import ast

tagging_list = ['NNG', 'NNP'] # 보통명사, 고유명사 태깅

def valid_token_choose(data):
    tagged_list = ast.literal_eval(data)
    tagged_dict = {word: pos for word, pos in tagged_list}
    selected_keys = [key for key, value in tagged_dict.items() if value in
tagging_list]
    return selected_keys

df['select_tagged'] = df.tk_text.apply(valid_token_choose)
```

출력 예시는 다음과 같다

```
[초등, 국, 영, 수, 전, 과목]
```

4. 최종 데이터 시각화(ongoing)

(ongiong) - 지호님 파트, 여건상 어려우면 빼기

IV. 사용한 알고리즘 설명

(ongoing) - 소미님 파트

1. MLP

2. Linear Regression

3. Random Forest

4. 비교

V. 알고리즘별 적합 및 결과

- Graphs, tables, any statistics (if any) 우리는 데이터에 적합한 모델을 탐색하기 위해 네 가지 모델을 적합해가며 비교하였다. 종속변수가 연속형 변수이기 때문에, 모델의 평가에는 RMSE(Root Mean Squared Error)와 R^2 score를 사용하였다.

1. MLP

MLP는 Keras 패키지를 사용해서 적합하였다.

1-(1). 데이터 전처리: 수치형, 범주형 데이터

```
np.random.seed(1) # numpy 랜덤시드 설정
tf.random.set_seed(0) # tensorflow 랜덤시드 설정

# 데이터 불러오기
data = pd.read_csv({file_path}, index_col=0)
data.drop(columns=['교습과목(반)', 'tk_text'], inplace=True)
X = data.drop(columns='교습비')
y = data.교습비
```

수치형 데이터는 `MinMaxScaler()`를 사용하여 0과 1 사이의 값으로 표준화하였고, 범주형 데이터는 원핫인코딩을 통해 더미변수로 확장하였다. 이는 MLP에서 입력층에 적용하기 위한 전처리 작업이다.

```
# 수치형 데이터 처리: 표준화
int_columns = ['총교습시간(분)', '강사수', 'ele_count',
               'mid_count', 'high_count', 'sch_count', 'pub_count',
               'priv_count',
               'ele_std_count', 'mid_std_count', 'high_std_count',
               'sch_std_count']
object_columns = ['분야구분', '교습계열', '교습과정', 'address_gu']

scaler = MinMaxScaler()
scaled_data = X[int_columns].values
scaled_data = scaler.fit_transform(scaled_data)
scaled_df = pd.DataFrame(scaled_data, columns=int_columns)

# 범주형 데이터 처리: 원핫인코딩
categorical_data = X[object_columns]
encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(categorical_data).toarray()
```

```
encoded_df = pd.DataFrame(encoded_data,
                           columns=encoder.get_feature_names(object_columns))
```

1-(2). 데이터 전처리: 토큰화 데이터 Padding과 Indexing

또한, 교습과목(반)에서 토큰화된 내용역시 padding과 indexing을 사용하여 리스트로 만들었다.

```
# 교습과목(반) 데이터 처리: 인덱싱, 패딩
tknz = Tokenizer()
tknz.fit_on_texts(X['tk_subject'].values)
sequences = tknz.texts_to_sequences(X['tk_subject'].values)
max_len = len(tknz.word_index) + 1
padded_sequences = pad_sequences(sequences, maxlen=max_len,
                                padding='post')
```

1-(3). 데이터 분할

처리된 데이터들을 모두 모은 후, train과 test 데이터로 분할하였다. 훈련용 데이터와 검증용 데이터를 8:2로 구성했다.

```
# 데이터 분할
X_concat = pd.concat([scaled_df, encoded_df], axis=1)
X_train, X_test, y_train, y_test, sequences_train, sequences_test =
train_test_split(
    X_concat, y, padded_sequences, test_size=0.2, random_state=1)
```

1-(4). 하이퍼파라미터 비교 및 최적화

그다음, 모형의 조율모수를 조정하면서 최적의 파라미터를 찾고자 했다.

```
# 하이퍼파라미터 튜닝 및 모델 학습
epochs_list = [50, 100]
batch_sizes = [16, 32, 64]
results = []

for epochs in epochs_list:
    for batch_size in batch_sizes:
        # MLP 모델 적합
        max_length = padded_sequences.shape[1]
        input_numeric = Input(shape=(scaled_df.shape[1]))
        input_categorical = Input(shape=(encoded_df.shape[1]))
        input_text = Input(shape=(max_length,))
        embedding_dim = 3

        embedding = Embedding(input_dim=max_len, output_dim=embedding_dim,
                              input_length=max_length)(input_text)
        flatten = Flatten()(embedding)
        concatenated = Concatenate()([input_numeric, input_categorical,
```

```

flatten])
    hidden = Dense(10, activation='relu')(concatenated)
    output = Dense(1)(hidden)
    model = Model(inputs=[input_numeric, input_categorical,
input_text], outputs=output)
    model.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mse'])
    model.summary()

    early_stopping = EarlyStopping(patience=5, monitor='mse',
restore_best_weights=True)
    history = model.fit(
        [X_train[scaled_df.columns], X_train[encoded_df.columns],
sequences_train], y_train,
        epochs=epochs, batch_size=batch_size, callbacks=
[early_stopping], verbose=2
    )

    # 모델 평가
    loss, mse = model.evaluate([X_test[scaled_df.columns],
X_test[encoded_df.columns], sequences_test], y_test)
    results.append((epochs, batch_size, loss, mse))

    print("Epochs:", epochs, "- Batch Size:", batch_size)
    print("Mean Squared Error (MSE):", mse)
    print("-----")

```

구성한 모델의 세부 사항은 다음과 같다

Layer (type)	Output Shape	Param #	Connected to
=====			
input_27 (InputLayer)	[(None, 2312)]	0	[]
embedding_8 (Embedding) ['input_27[0][0]']	(None, 2312, 3)	6936	
input_25 (InputLayer)	[(None, 12)]	0	[]
input_26 (InputLayer)	[(None, 91)]	0	[]
flatten_8 (Flatten) ['embedding_8[0][0]']	(None, 6936)	0	
concatenate_8 (Concatenate) ['input_25[0][0]', 'input_26[0][0]',	(None, 7039)	0	

```
'flatten_8[0][0]'
```

```
dense_16 (Dense) (None, 10) 70400
```

```
['concatenate_8[0][0]']
```

```
dense_17 (Dense) (None, 1) 11
```

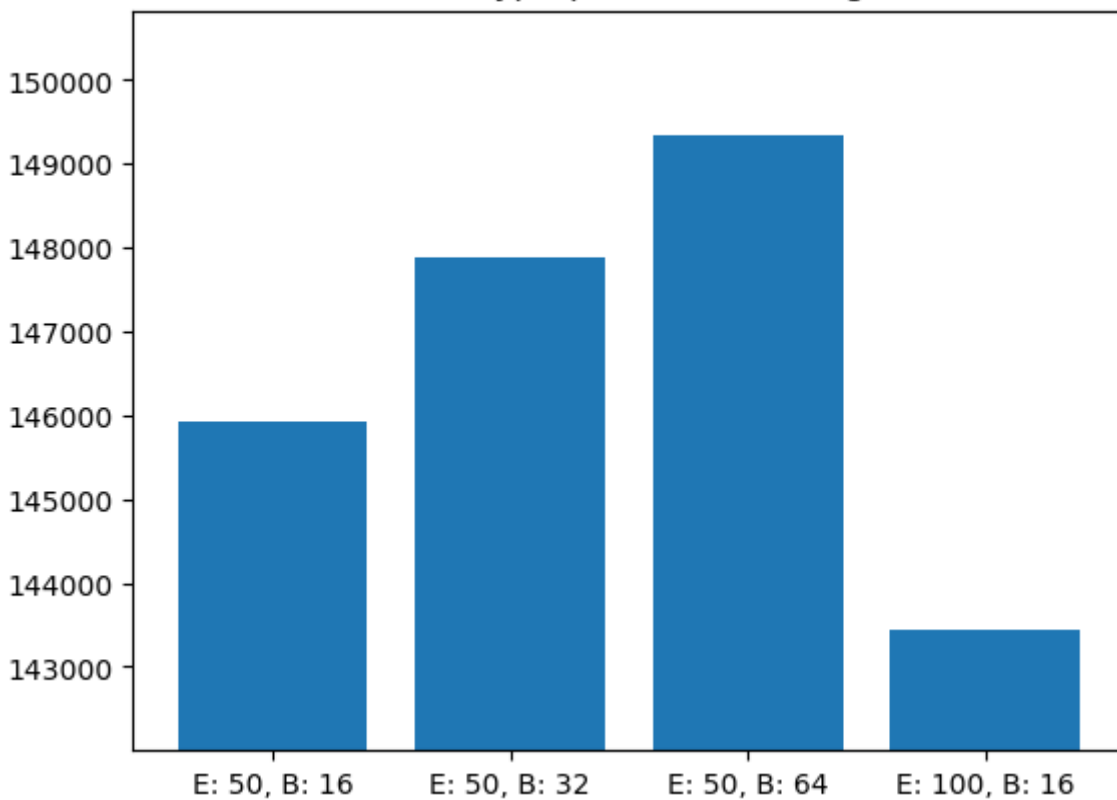
```
['dense_16[0][0]']
```

```
=====  
Total params: 77,347
```

```
Trainable params: 77,347
```

팀원이 보유한 컴퓨터의 컴퓨팅 파워의 한계로 총 4가지의 경우를 적합, RMSE를 비교하였다. 비교한 그래프는 다음과 같다.

MLP Hyperparameter Tuning



가장 낮은 RMSE를 보인 값은 Epochs = 100, Batch_Size = 16이었다. 따라서 이 값을 하이퍼 파라미터로 사용하기로 한다.

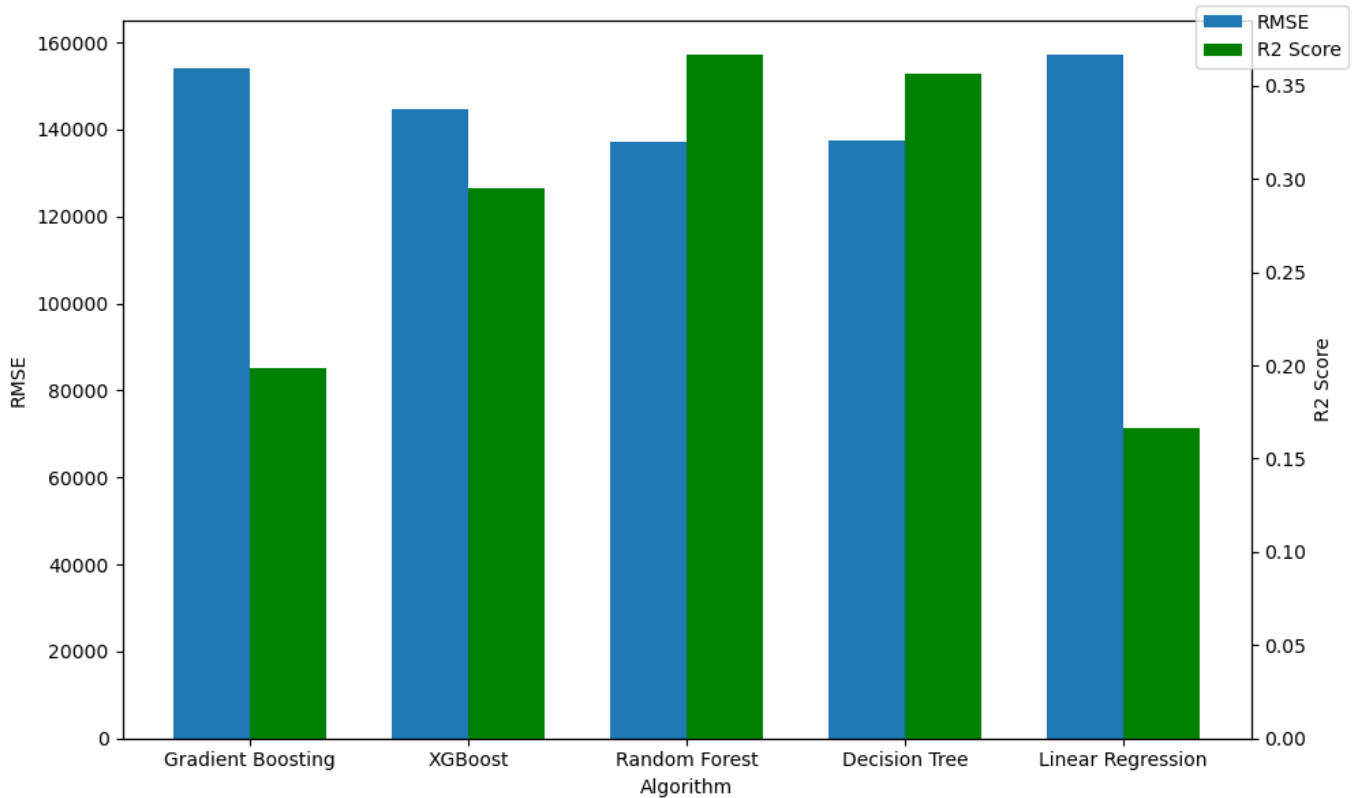
2. Linear Regression

3. Regression Tree

4. Random Forest

5. XGboost

0. 모델 비교

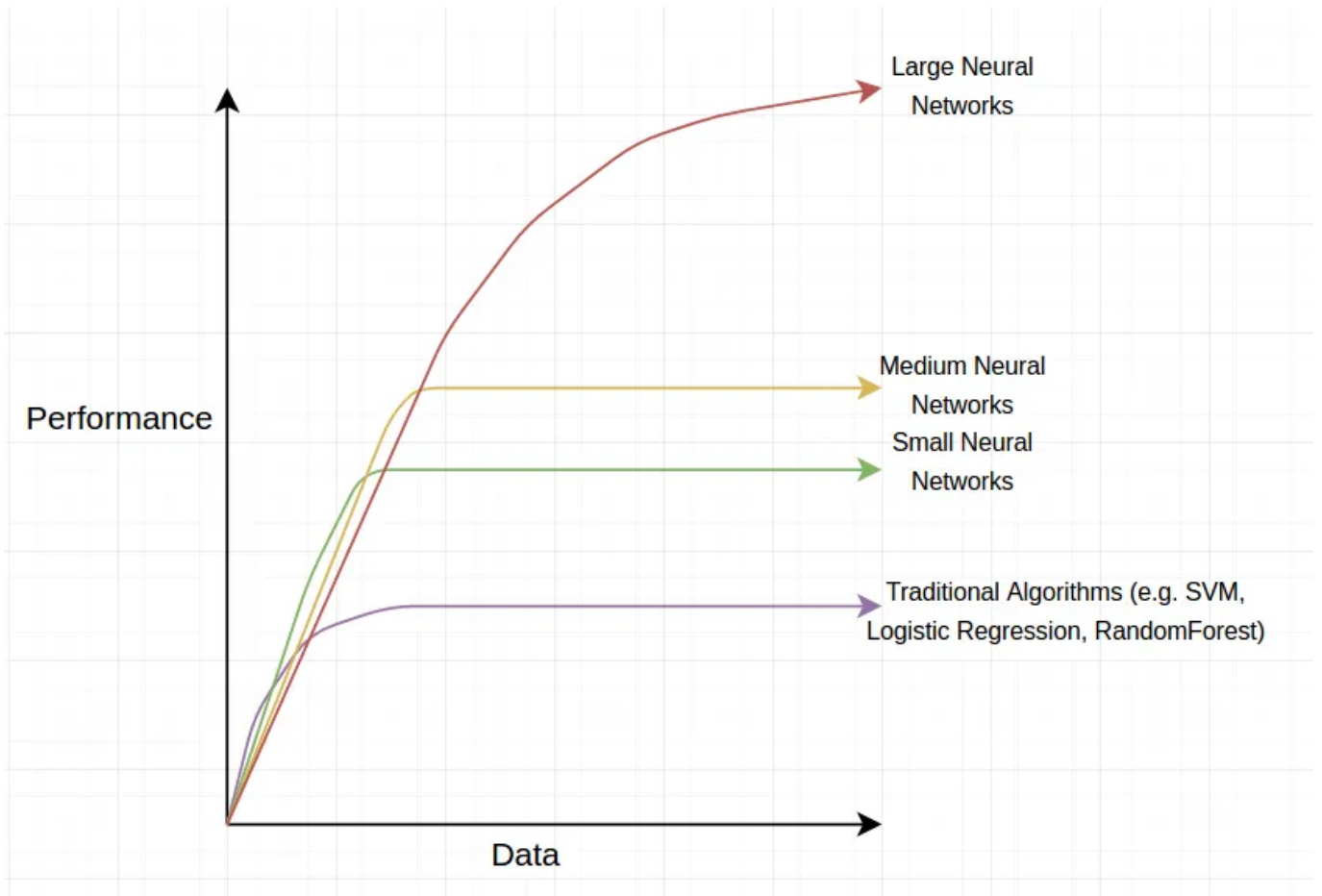


VI. 결론 및 논의

본 팀보고서에서는 서울시교육청에서 제공한 공공데이터인 학원별 교습비용 데이터를 통해 학원을 예측하는 모델을 만들어 보았다. 지오코딩을 활용해 학교 데이터를 변수로 포함시켰고, 형태소분석기를 통해 학원별로 다르게 작성되어 있는 교습과목의 데이터에서 유효한 데이터를 추출, 가공하였다. 전처리된 데이터를 바탕으로 다양한 모형에 적합해보면서 어떤 모형이 가장 잘 예측하는지, 사교육 비용을 예측하는데 가장 적절한 모형이 어떤 것인지 찾고 선택하였다.

1. Multi-Layer Perceptron vs. Traditional Machine Learning

MLP와 기존의 머신러닝 알고리즘을 모두 적합해 본 결과, 우리가 가지고 있는 데이터셋에서는 MLP보다 기존의 ML 알고리즘이 더 작은 test validation error를 보여주었다. 이는 MLP가 모든 ML 알고리즘보다 뛰어나다는 것은 아니며, 분석가의 데이터 가공과 적절한 조율모수의 선택이 기존의 ML에서 훨씬 더 좋은 결과를 가져올 수 있다는 점을 시사한다.




또한, MLP가 최대한의 성능을 내기 위한 데이터가 존재한다는 사실도 알 수 있었다. 특히, 딥러닝 알고리즘은 학습 데이터의 크기가 압도적으로 클 때 효과적으로 작동한다(James et al. 2021). 하지만, 이번 분석에서 사용한 데이터의 표본 수는 약 13만개이다. 더 나아가, 변수가 다양하지 못했다는 점, 한 학원에서 여러 교습과목을 가지고 있기 때문에, 학원 고유의 데이터에 기반한 변수간의 상호작용이 존재했다는 점 등이 학습에서 실질적으로 유용한 데이터의 수를 훨씬 줄였을 것으로 예상된다.

2.

VII. 출처 및 사용 패키지

본 보고서에 사용한 라이브러리와 참고문헌은 다음과 같다.

1. 라이브러리

- pandas: 데이터 처리
- numpy: 데이터 계산
- urllib, json: 네이버 지도 API 사용
- konlpy.tag.Kkma: 텍스트 데이터 토큰화  꼬꼬마 홈페이지
- ast: 토큰화 전처리
- matplotlib.pyplot: 시각화
- tensorflow, keras: MLP 모형 적합

2. 참고문헌

- [BallTree와 거리 계산](#)

- [네이버 지도 API](#)
- [네이버 API를 이용한 지오코딩\(Geocoding\)by Kim](#)
- 장철원(2021). 선형대수와 통계학으로 배우는 머신러닝. BJ퍼블릭.
- James et al.(2021). An Introduction to Statistical Learning with Application in R
-

각주

- 1: 교육부(2023). 2022 교육부 초등고교 사교육비 조사
- 2: 네이버 클라우드 플랫폼 [지오코딩 API 설명서](#)
- 3: 꼬꼬마 형태소 분석기 [태그 리스트](#)