심장질환 AI 예측서비스

(우병목)

<table>
<tr><td>프로젝트명</td><td colspan="3">심장질환 AI 예측서비스</td></tr>
<tr><td>프로젝트 주제</td><td colspan="3">심장질환 예측 머신러닝 모델링 및 검증, 시각화</td></tr>
<tr><td>프로젝트 기간</td><td colspan="3">2023. 02. 16 ~ 2023. 02. 17</td></tr>
<tr><td rowspan="6">팀원 담당업무
및 기여사항</td><td>이름</td><td>김경목</td><td></td></tr>
<tr><td colspan="3">시각화, 머신러닝 학습, 머신러닝 모델링</td></tr>
<tr><td>이름</td><td>우상욱</td><td></td></tr>
<tr><td colspan="3">EDA, 머신러닝 모델링, 전처리, 웹 구현</td></tr>
<tr><td>이름</td><td>민병창</td><td></td></tr>
<tr><td colspan="3">전처리, 머신러닝 학습 및 선택, 시각화, 머신러닝 모델링</td></tr>
<tr><td>구분</td><td colspan="3">내용</td></tr>
<tr><td>프로젝트 목적
및 동기</td><td colspan="3">범용성있는 결과가 나오는 프로젝트를 진행하는 편이 필요하다고 판단했다. 질병과 관련된 데이터는 많은 사람들이 관심을 가진다. 때문에 심장질환을 예측할 수 있는 Kaggle안의 데이터를 보고 프로젝트를 진행하게 되었다</td></tr>
<tr><td>개발환경</td><td colspan="3">개발환경 : Python 3.9.13
라이브러리
- 데이터 수집 및 전처리 : pandas, numpy
- 시각화 : matplotlib , Seaborn
- 머신러닝 : scikit-learn, lightgbm , xgboost ,catboost
- 웹구현 : streamlit
Devops
- Git, Notion, Canva</td></tr>
</table>

차 례

Ⅰ. 서　　론

1. 프로젝트 배경

CDC, 미국 질병관리위원회에서 2020년 40만명의 성인을 대상으로 전화 설문을 통해 심장질환 유무를 수집한 데이터를 캐글에서 발견하였다. 일단 이 데이터는 기본적으로 전처리가 되어있는 데이터였고, 데이터 수집을 하는데에는 그렇게 큰 어려움은 없었다. 이 데이터를 활용해서 머신러닝 모델을 만들고, 이 모델을 활용해서 일반인들이 쉽게 자신의 심장질환을 예측해볼 수 있는 웹을 구현하기로 하였다.

2. 프로젝트 목표

데이터를 가지고 심장질환을 예측하는 머신러닝을 모델링하고 정확도와 재현율을 목표한 수치만큼 올리는 과정을 통하여 이제까지 공부한 내용들을 정리하고 직접 경험한다. 그리고 만들어진 머신러닝 모델로 심장질환을 예측할 수 있는 웹을 구현한다

Ⅱ. 방 법 및 내 용

1) 전처리

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline

import re
import warnings
warnings.filterwarnings('ignore')
```

```
# preprocessing
df = pd.read_csv('./../0.data/heart_2020_cleaned.csv')
# replace 를 위한 함수 dataframe, column = 바꾸고자 하는 열 , text = 바꿀 문자열을 리스트로 넣어주면 list개수만큼 숫자로 변환해준다.
df
```

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealt
0	No	16.60	Yes	No	No	3.0	30.0	No	Female	55-59	White	Yes	Yes	Very goo
1	No	20.34	No	No	Yes	0.0	0.0	No	Female	80 or older	White	No	Yes	Very goo
2	No	26.58	Yes	No	No	20.0	30.0	No	Male	65-69	White	Yes	Yes	Fa
3	No	24.21	No	No	No	0.0	0.0	No	Female	75-79	White	No	No	Goo
4	No	23.71	No	No	No	28.0	0.0	Yes	Female	40-44	White	No	Yes	Very goo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
319790	Yes	27.41	Yes	No	No	7.0	0.0	Yes	Male	60-64	Hispanic	Yes	No	Fa
319791	No	29.84	Yes	No	No	0.0	0.0	No	Male	35-39	Hispanic	No	Yes	Very goo
319792	No	24.24	No	No	No	0.0	0.0	No	Female	45-49	Hispanic	No	Yes	Goo
319793	No	32.81	No	No	No	0.0	0.0	No	Female	25-29	Hispanic	No	No	Goo

```
from sklearn.preprocessing import LabelEncoder

### 라벨 인코딩
# Yes or No 로 대답한 데이터들과 Female or Male 로 대답한 값에 대해 LabelEncoding을 진행함.
# Diabetic은 예외적으로 4가지 경우에 대해서 물라와있었기에 0,1,2,3으로 카테고리화함
label_features = ['HeartDisease', 'Smoking', 'AlcoholDrinking', 'Stroke',
                  'DiffWalking', 'Sex', 'PhysicalActivity','Asthma',
                  'GenHealth', 'Diabetic', 'KidneyDisease', 'SkinCancer']

for feature in label_features:
    encoder = LabelEncoder()
    item = df[feature].unique()
    encoder.fit(item)
    df[feature] = encoder.transform(df[feature])

# 나이와 인종에 대해서는 원핫 인코딩을 진행함
onehot_features = ['AgeCategory', 'Race']
df = pd.get_dummies(df, columns= onehot_features)
```

```
# 중복행 제거
df = df.drop_duplicates()
```

```
df = df.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
df
```

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	Diabetic	...	AgeCategory_9	AgeCategory_10	AgeCategory_1
0	0	16.60	1	0	0	3.0	30.0	0	0	1	...	0	0	
1	0	20.34	0	0	1	0.0	0.0	0	0	0	...	0	0	
2	0	26.58	1	0	0	20.0	30.0	0	1	1	...	1	0	
3	0	24.21	0	0	0	0.0	0.0	0	0	0		0	0	

```
#이상치 데이터를 제거하려 했으나 일단 놔두기로함.
import numpy as np

def remove_outlier(DF = None, columns = None, weight=1.5):
    # 입력받은 dataframe과 columns에 대해 이상치가 없는 데이터를 반환하는 함수
    target = DF[columns]
    quant_25 = np.percentile(target.values, 25)
    quant_75 = np.percentile(target.values, 75)
    # IQR을 계산한다.
    IQR = quant_75 - quant_25
    # IQR에 1.5를 곱해서 최대 최소 지점을 구한다.
    W = IQR * weight
    lowest = quant_25 - W
    highest = quant_75 + W
    print(highest , lowest)
    # 최대값 보다 크거나, 최소값 보다 작은 값을 아웃라이어로 설정하고 DataFrame index 반환.
    not_outlier_index = target[(target >= lowest) & (target <= highest)].index
    return not_outlier_index
#Sleep time 의 이상치를 제거함(12시간 이상 자는 사람 + 2시간 이하로 자는 사람 )
#df = df.iloc[remove_outlier(df,'SleepTime')] # 일단 스킵
```

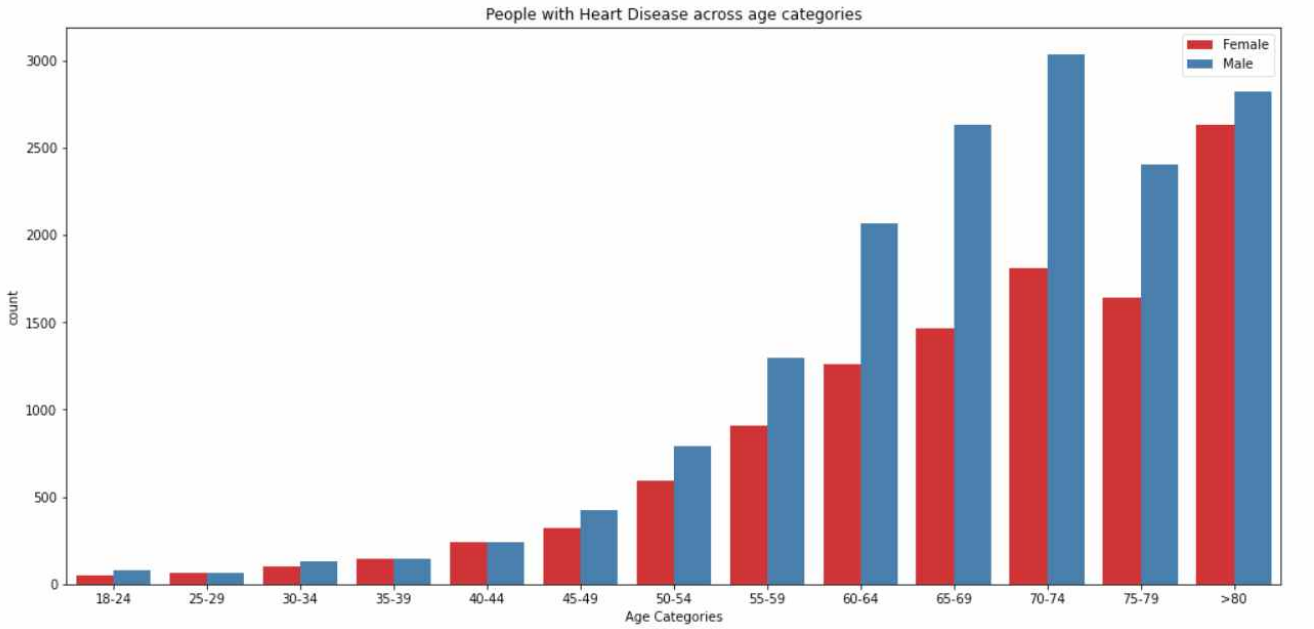
데이터에 결측치가 없어서 별도로 처리할 것은 없었다 '예', '아니오'와 '남성','여성'의 데이터는 라벨 인코딩으로 데이터를 변환하였다 나이와 인종데이터는 원-핫 인코딩을 진행하였고 중복행을 제거하고 나서 데이터프레임의 행의 개수는 319,795개에서 301,717개로 줄어든 것을 볼 수 있었다

2) 시각화

심장질환이 있는 사람의 범주형 변수 시각화

```
Heart_O_df = df2[df2['HeartDisease'] == 'Yes']
cat_vis(Heart_O_df)
```

```
df2.loc[df2.AgeCategory=='80 or older','AgeCategory'] = '>80'
order = pd.unique(df2.AgeCategory.values)
order.sort()
plt.figure(figsize = (17,8))
sns.countplot(data=df2[df2['HeartDisease']=='Yes'], x='AgeCategory', hue='Sex', palette='Set1', order=order)
plt.title('People with Heart Disease across age categories')
plt.xlabel('Age Categories')
plt.legend(['Female','Male'])
plt.show()
```



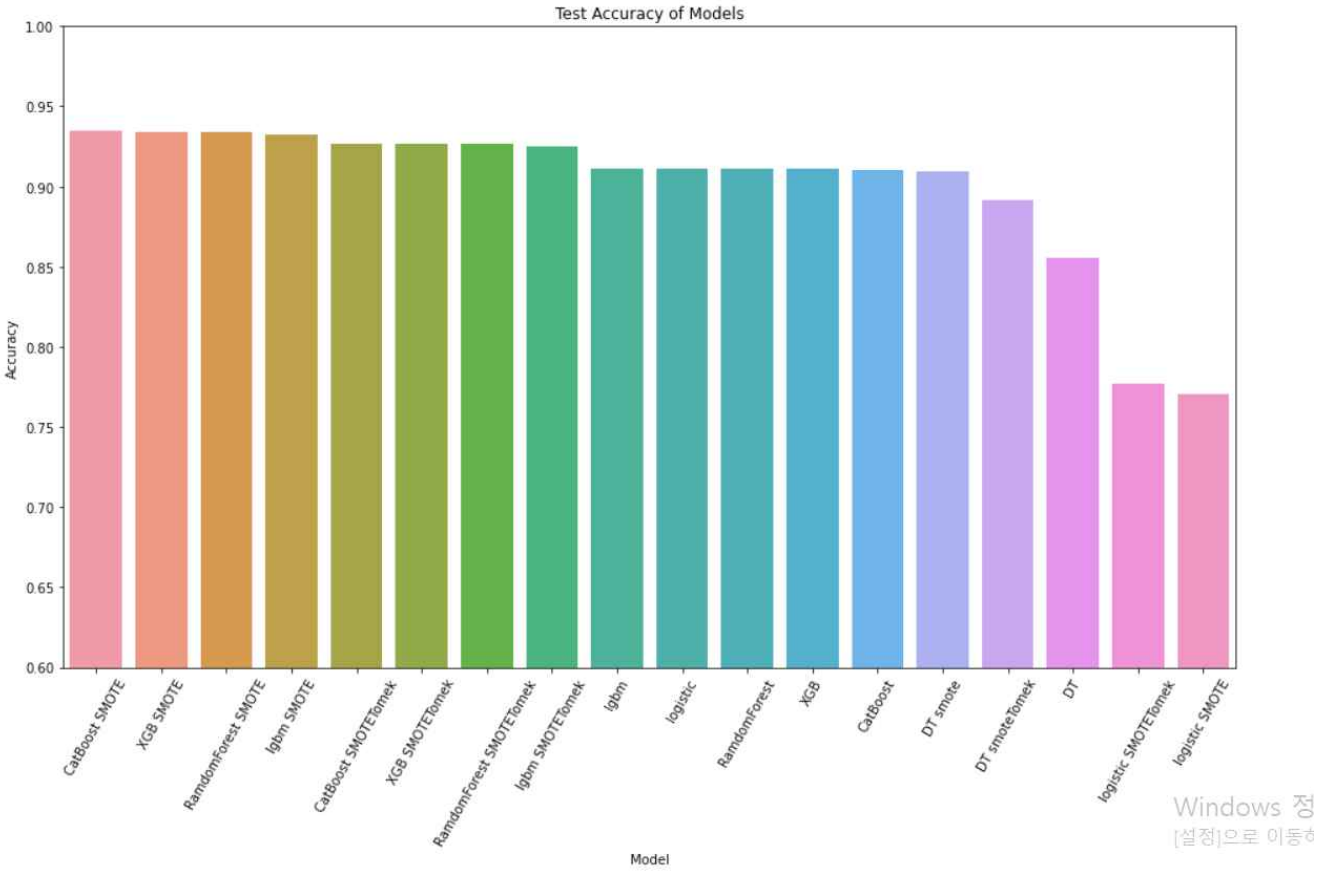
그래프는 seaborn 라이브러리를 사용했으며 countplot을 이용하여 나타내었다. 심장질환을 가진 사람에 대해서 막대그래프를 만들고 성별에 따라서 카테고리를 분류했다, 심장질환을 가진 사람은 나이가 많고 남자일수록 더 많은 것으로 보인다

연속형 변수 상관관계 시각화

```
colormap = plt.cm.PuBu
plt.figure(figsize=(16,9))
plt.title("Person Correlation of Features", y = 1.05, size = 15)
sns.heatmap(df2.corr(), linewidths = 0.1, vmax = 1.0,
            square = True, cmap = colormap, linecolor = "white", annot = True, annot_kws = {"size" : 10})
plt.show()
```

그래프는 seaborn라이브러리의 heatmap을 이용하여 상관관계를 직관적으로 표현하려고 하였다. 그리고 수치형 변수를 시각화 하였다. 수치형 변수간의 상관관계를 나타내 보았고 .변수 간의 유의하다고 생각할만한 상관관계가 발견되지 않았다. 이를 통해서 수치형 변수 간 다중공선성에 대한 의심은 일단 배제를 했다.

3) 모델링



그리드 서치 결과 kfold 검증

결과(random_state = 42, task_type = 'GPU')

- bagging_temperature = 0
- depth = 9
- l2_leaf_reg = 3
- learning_rate =0.1

```
# CatBoost 모델을 만듭니다.
catboost = CatBoostClassifier(random_state = 42,verbose = 1000, bagging_temperature = 0, depth = 9, l2_leaf_reg = 3, learning_rate =0.1)#, task_type="GPU")
fold_K(X_smote, y_smote, catboost)
```

```
0:      learn: 0.5920657        total: 112ms    remaining: 1m 52s
999:    learn: 0.1222221        total: 2m 7s    remaining: 0us
0:      learn: 0.5919828        total: 108ms    remaining: 1m 48s
999:    learn: 0.1223949        total: 2m 15s   remaining: 0us
0:      learn: 0.5921759        total: 169ms    remaining: 2m 48s
999:    learn: 0.1221722        total: 2m 25s   remaining: 0us
0:      learn: 0.5926977        total: 160ms    remaining: 2m 39s
999:    learn: 0.1216345        total: 2m 25s   remaining: 0us
0:      learn: 0.5920587        total: 148ms    remaining: 2m 27s
999:    learn: 0.1230085        total: 2m 36s   remaining: 0us
정확도 : train score : 0.952345731077531
정확도 : test score : 0.9360771865710775
재현율 : train score : 0.9219948526638477
재현율 : test score : 0.9009552166828586
<catboost.core.CatBoostClassifier at 0x28b806bc5e0>
```

그리드 서치 결과 KFold 검증

결과(random_state = 42)

- colsample_bytree = 0.8,
- learning_rate = 0.1,
- max_depth = 5,
- min_child_weight = 4,
- n_estimators = 200,
- subsample = 0.9

```
xgb = XGBClassifier(random_state = 42)
param_grid = {
    'colsample_bytree' : [0.8],
     'learning_rate' : [0.1],
     'max_depth' : [5],
     'min_child_weight' : [4],
     'n_estimators' : [200],
     'subsample' : [0.9] }

grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=3)
grid_search.fit(X_smote, y_smote)
fold_K(X_smote, y_smote, xgb)
```

```
정확도 : train score : 0.9366433051647263
정확도 : test score : 0.9342426484002667
재현율 : train score : 0.8970401234704963
재현율 : test score : 0.894614708024282
```

30만개의 기존데이터, SMOTE 기법으로 처리하여 약 55만개로 오버샘플링된 데이터 그리고 언더샘플링과 오버샘플링을 동시에 진행하는 SMOTEOMEK기법으로 처리하여 약 25만개의 데이터로 샘플링된 3가지 데이터를 각 모델에 넣고 비교를 해보았다 정확도 지표와 과적합 여부를 판단하고 가장 상위모델은 SMOTE를 적용한 catboost모델과 xgboost모델을 선정하였다 이 두 모델을 하이퍼파라미터 튜닝을 한 결과 catboost에서 test데이터의 정확도가 0.0032정도 증가했고 재현율은 0.0086 증가하였다 xbgoost는 test 데이터의 정확도가 0.0093 증가했고 재현율은 0.0109 증가하였다

Ⅳ.결　　론

1. 결론

본 프로젝트에서는 데이터를 전처리와 머신러닝 모델의 정확도와 재현율의 증가를 위해 여러 가지 모델을 적용시켜보고 상위 모델 2개에 하이퍼 파라미터 튜닝을 진행하였다 사용자 정보를 입력받아서 최종 선택된 모델에 적용시켜 예측값을 반환시켜주는 웹을 구현하였고 모바일환경에서 원활하게 구동되며 예측값이 나타나는 것을 확인하였다.