Stroke Prediction Dataset

11 clinical features for predicting stroke events

Graduate Admission

Week9 : Decision Tree

Data Mining – 이원상 교수님
실습 9주차

김서연 김연재 윤수진 이가람

I. Classification

II. Regression

I. Classification

1. Intro

현업에서 사용하는 데이터에는 결측치를 갖거나 사용하기 어려운 형태를 갖는 데이터들이 포함되어 있다. 이러한 데이터들은 분석 시 데이터의 품질을 떨어뜨리고 분석의 효용을 낮춘다. 따라서 데이터 분석과 모델 학습에 사용되는 데이터의 형태로 데이터를 가공하는 전처리 과정은 반드시 수반되어야 한다. 본 실습에서는 우리 조가 선택한 주제에 대해 research question을 제시하고, 이를 답하는데 필요한 데이터 분석 과정에 적절한 전처리 과정을 적용하려고 한다.

세계보건기구(WHO)에 따르면 뇌졸중은 전 세계 사망 원인 2위인 치명적인 질병이다. 급작스럽게 찾아오는 응급질환인 만큼 발병 시기를 예측하기도 어렵다. 그 중 고혈압이 있거나, 당뇨병 환자이거나, 흡연을 한다면 그 위험이 더욱 커진다. 실제로 하루 40개비 이상의 담배를 피우는 사람은 10개비 이하의 담배를 피우는 사람에 비해 뇌졸중에 걸릴 가능성이 2배 더 높다. 이는 흡연이 기저질환이 있는 환자의 뇌혈관 손상을 가속화시키고 뇌경색의 가능성을 더욱 높이는 요인이 되기 때문이다. 그렇다면 하루 40개비 이상의 담배를 피우는 흡연자가 10개비 이하의 담배를 피우는 비흡연자에 비해 뇌졸중 진단 가능성이 높다는 가설은 타당한가? 심장병등의 기저질환이 있는 환자의 경우 뇌졸중에 걸릴 가능성이 더 높은가? 흡연 여부 이외에 뇌졸중에 영향을 미치는 요소가 존재하는가?

환자들의 데이터를 분석해 뇌졸중의 발병 여부를 예측한다면 뇌졸중이 찾아올 공포를 줄이는 것은 물론, 발병하지 않도록 더욱 철저히 예방까지 할 수 있을 것이다. 본 실습 과정은 이 데이터 세트의 성별, 나이, 각종 질병, 흡연 상태와 같은 입력 매개변수를 전처리하고, 각 데이터의 분포를 확인한다. 나아가 최종적으로 각 요소가 뇌졸중에 미치는 영향을 확인한다.

2. Research Question

2.1. 흡연자의 경우, 비흡연자에 비해 뇌졸중에 걸릴 가능성이 상대적으로 높은가?
2.2. 기저질환이 있을 경우 뇌졸중에 걸릴 가능성이 높은가?
2.3. 성별, 나이, 각종 질병 여부, 거주 지역이나 결혼 여부 등의 생활양식을 통해 뇌졸중의 예측이 가능한가?

3. Dataset

3.1 Data Explanation

Research Question에 답하기 위하여 'Kaggle'에서 제공하는 'Stroke Prediction Dataset'[1]을 사용하였다. 해당 데이터셋은 IEEE 저널에 실린 논문, 'Identifying Stroke Indicators Using Rough Sets'[2]에 의하면 McKinsey & Company가 관리하는 Electronic Health Record(EHR)을 기반으로 두고 있다. 분석에 활용한 이 데이터셋은 5110x12의 2차원 tabular data의 형태를 띄고 있다. 각각의 속성은 stroke에 영향을 끼칠만한 요소들로 구성되어 있는데, 자세한 속성 정보는 다음과 같다.

<Attribute Information>

1) id: 환자 분별 id
2) gender: 성별, "Male", "Female", "Other"
3) age: 환자의 나이
4) hypertension: 환자에게 고혈압이 없는 경우 0, 있는 경우 1
5) heart_disease: 환자에게 심장병이 없는 경우 0, 있는 경우 1
6) ever_married: 결혼 여부, "No", "Yes"
7) work_type: 고용 상태, "children", "Govt_jov", "Never_worked", "Private", "Self-employed"
8) Residence_type: 거주 지역, "Rural", "Urban"
9) avg_glucose_level: 평균 혈중 글루코스(당) 수치
10) bmi: 체질량 지수
11) smoking_status: 흡연 상태, "formerly smoked", "never smoked", "smokes", "Unknown"*
12) stroke: 환자에게 뇌졸중이 있었을 경우 1, 아닌경우 0

*Note: smoking_status 컬럼에서 환자의 상태를 알 수 없을 경우 "Unknown"으로 표기하였다.

```
[ ] df.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Female	61	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	Female	79	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

[1] https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

[2] Pathan, Muhammad Salman & Zhang, Jianbiao & John, Deepu & Nag, Avishek & Dev, Soumyabrata. (2020). Identifying Stroke Indicators Using Rough Sets. IEEE Access. 8. 10.1109/ACCESS.2020.3039439.

각 속성의 실제 데이터 타입을 확인해보았을 때, gender, ever_married, work_type, Residence_type, smoking_status는 object(string), 나머지의 경우 float과 int등 수치형 데이터가 혼합되어있었다. 하지만 의미적으로 수치형이 아닌 범주형으로 분류 되어야하는 hypertension, heart_disease, stroke는 모두 string타입으로 변경해주었고, age 컬럼의 경우 1세 이하의 영유아에 대해서만 소수점을 가지고 나머지의 경우 int형식을 가지고있었기 때문에, 전반적으로 소수점을 버림하여 int로 통일하였다.

```
[ ]  df.dtypes

     id                   int64
     gender              object
     age                float64
     hypertension         int64
     heart_disease        int64
     ever_married        object
     work_type           object
     Residence_type      object
     avg_glucose_level  float64
     bmi                float64
     smoking_status      object
     stroke               int64
     dtype: object
```

```
[ ]  df['age'] = df['age'].apply(lambda x: int(x))
     df.drop('id', axis=1, inplace=True)
     for i in ['gender','hypertension', 'heart_disease', 'ever_married',
            'work_type', 'Residence_type', 'smoking_status', 'stroke']:
       df[i] = df[i].apply(lambda x: str(x))
```

3.2 EDA

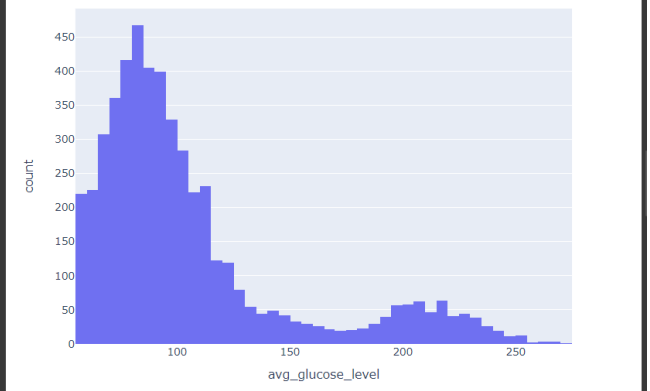
앞서 제시한 Research Question과 관련하여, 각 변수의 분포확인이나 stroke와 다른 변수간의 상관관계 확인, 추가 전처리 아이디어를 얻기 위한 EDA를 진행하였다. EDA를 위한 시각화 라이브러리로는 seaborn, matplotlib 그리고 plotly를 활용하였다.

3.2.1 수치형 변수

수치형 변수에 대해서는 히스토그램으로 그 분포를 확인해 보았다.

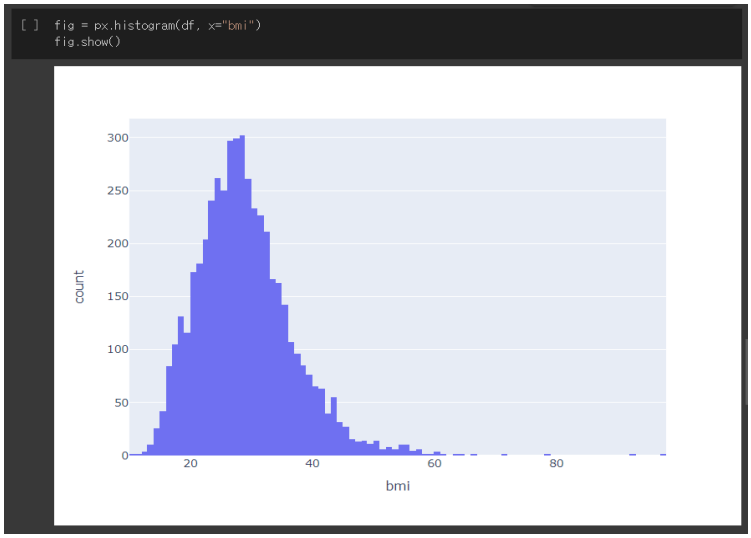
A. avg_glucose_level의 히스토그램 : 정규분포 봉우리가 두개가 보이는 Gaussian Mixture Model로 보거나 왼쪽으로 치우친 분포라고 볼 수 있다. 특별히 이상치라고 볼만한 수치는 육안으로는 찾기 어려웠다.

```
[ ]  fig = px.histogram(df, x="avg_glucose_level")
     fig.show()
```

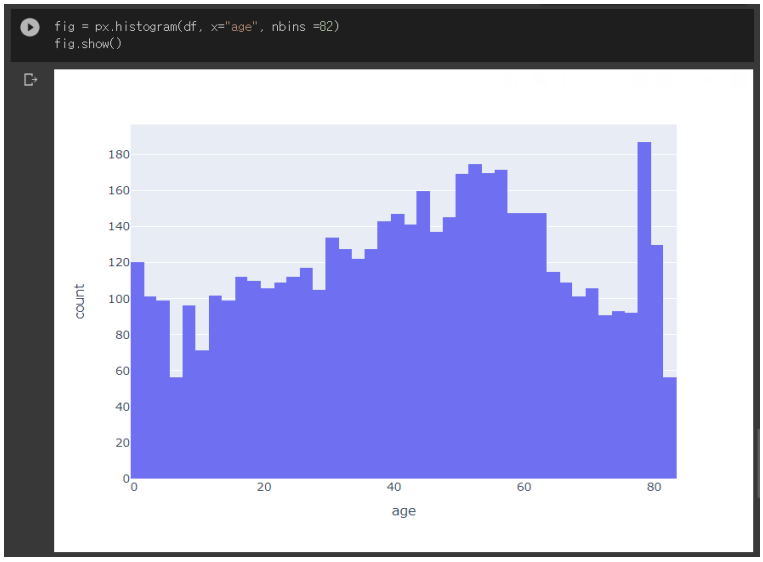


B. bmi의 히스토그램 : 정규분포와 매우 유사하게 보인다. 하지만 이상치로 보이는 레코드가 몇 건 있을 것으로 예상되어 box plot을 통한 확인을 거친 후 결과에 따라 이상치에 민감한 min-max scaling은 지양하는 것이 올바른 방향으로 보인다.

```
[ ]  fig = px.histogram(df, x="bmi")
     fig.show()
```



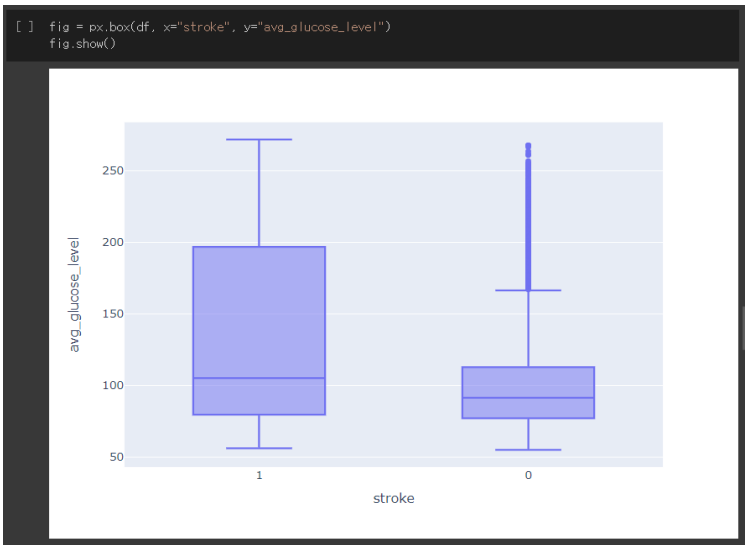
C. age의 히스토그램 : 1세 이하의 영유아에게서 0.8과 같은 float형 데이터가 있음을 확인하였다. 이를 모두 소수점 첫째자리 버림하여 age를 interger로 통일한 후 히스토그램을 확인해보았다. 다른 수치형 변수에 비해 비교적 고른 분포를 보이는 것을 확인할 수 있었다. 특별한 이상치 또한 확인할 수 없었다.



3.2.2 수치형 변수의 stroke과의 상관관계

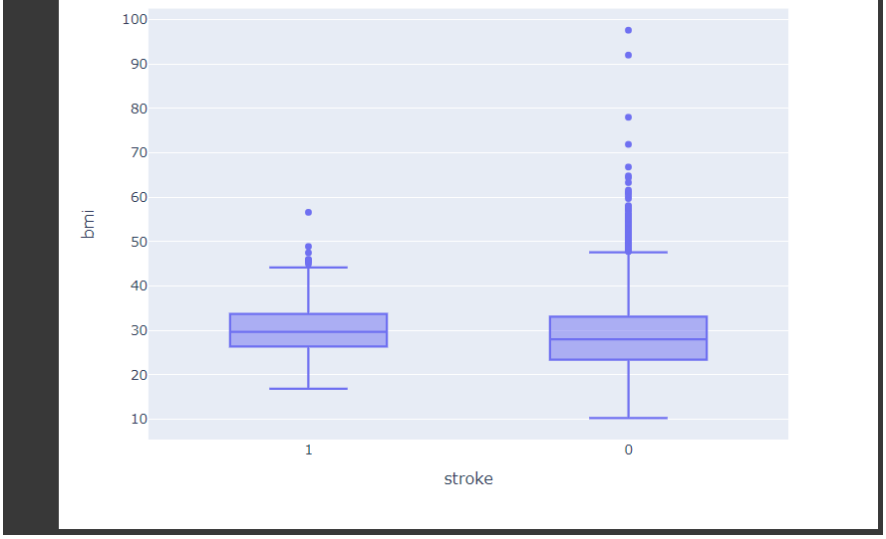
다음으로는 예측해야할 label인 stroke를 기준으로 각 수치형 분포를 boxplot으로 시각화해보았다.

A. avg_glucose_level : stroke여부에 따른 avg_glucose_level의 분포 보았을때 두 분포의 중앙값과 q3가 크게 다른 것을 확인할 수 있었다. 이에 avg_glucose_level은 stroke 예측/분류에 좋은 변수라고 판단할 수 있다.



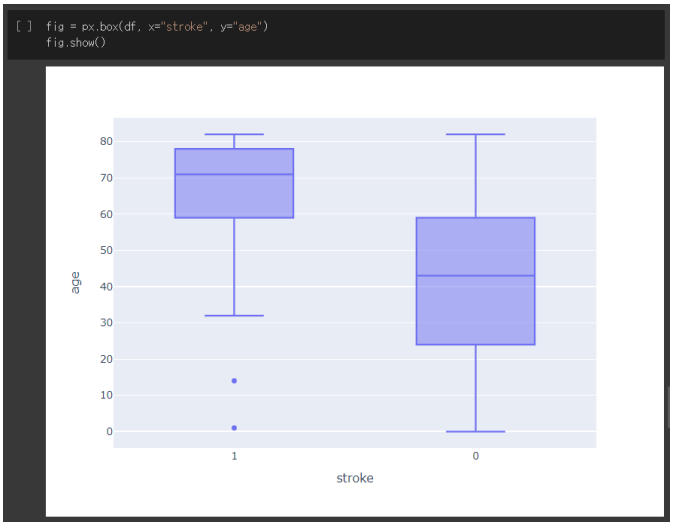
B. bmi : stroke여부에 따른 bmi의 분포 보았을때 0의 경우 표본의 수가 많아 넓게 퍼져있는 반면 1의 경우 비교적 좁게 분포하였다. 중앙값 및 q1,q2,q3가 크게 다르지 않아 bmi의 stroke과의 상관관계를 추가로 확인해야할 필요가 있다.

```
[ ]  fig = px.box(df, x="stroke", y="bmi")
     fig.show()
```



C. age : 반면 나이의 경우 stroke와 큰 연관성이 있다고 볼 수 있었다. stroke 0에서는 나이가 전반적으로 고르게 분포하는 반면, 1에서는 0~30세 사이에 분포하는 건수는 2건에 불과했고 나머지는 모두 30대 이상의 나이를 가지는 것을 확인하였다.

```
[ ]  fig = px.box(df, x="stroke", y="age")
     fig.show()
```

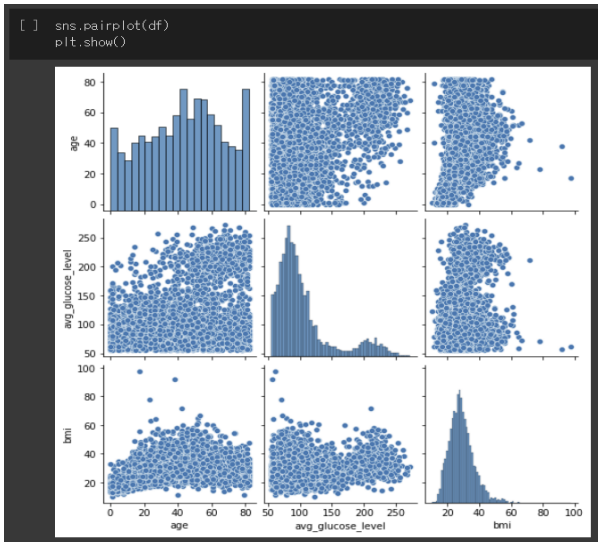


추가로 수치형 변수 간 pearson 상관계수도 확인해보았다. 절댓값 0.3~0.4이상을 상관관계가 크다고 이야기할 수 있는데, 눈에띄게 큰 상관관계를 가지는 수치형 변수는 확인하기 어려웠다.

```
[ ]  df.corr()
```

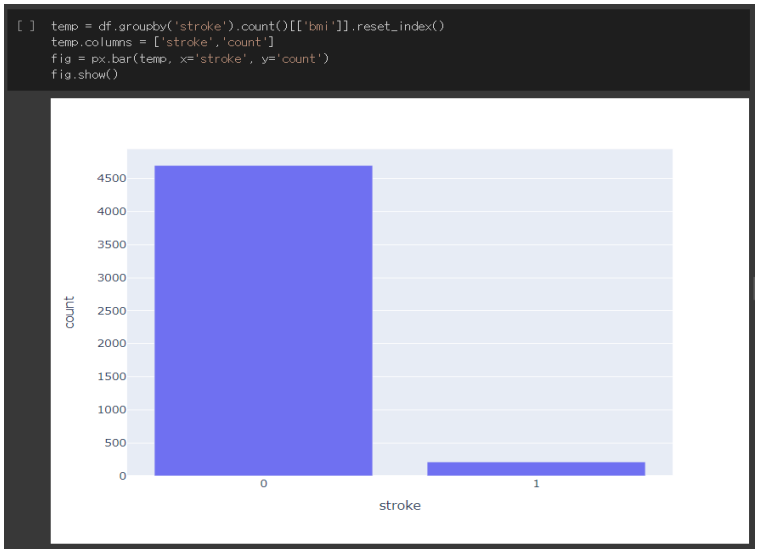
	age	avg_glucose_level	bmi
age	1.000000	0.238060	0.333738
avg_glucose_level	0.238060	1.000000	0.175502
bmi	0.333738	0.175502	1.000000

마지막으로 모든 수치형 변수의 각 분포와 변수간 산점도를 다음과 같이 한번에 확인할 수 있다.

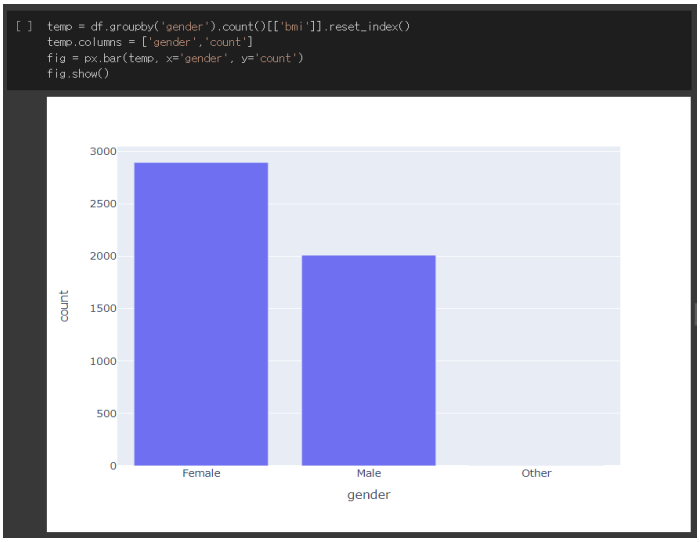


3.2.3 범주형 변수

A. Target인 stroke의 label 분포 : 0인 레코드 수는 4861, 1인 레코드 수는 249로 1인 레코드가 전체의 4.9%인 것을 확인할 수 있었다. 우리의 research question 중 stroke의 예측/분류 문제를 풀기 위해서는 이러한 imbalanced dataset을 그대로 활용할 경우 모델이 데이터를 제대로 학습하지 못하는 문제가 발생할 수 있다. 이러한 문제를 방지하기 위해 Undersampling이나 Oversampling 과정을 거쳐야 한다.



B. gender의 분포 : 전체 5110건 중 2994건이 여성, 2115건이 남성, 1건이 other인 것을 확인하였다. other의 경우 전체에서의 비율이 매우 적고 의미적으로 모호하므로 추후 해당 레코드를 제거할 필요가 있다.



C. 나머지 범주형 데이터에 대해서는 한번에 각 라벨의 수를 barplot으로 시각화해보았다. 그 결과 고혈압 여부, 심장병 발생 여부에 대해서는 0,1의 비율이 크게 다른것을 확인할 수 있었다. 그 외의 경우 대부분의 라벨이 비교적 고르게 분포하였다. 각 범주형 데이터에 대해 라벨의 종류가 2가지인 경우 0,1로 인코딩할 수 있다.

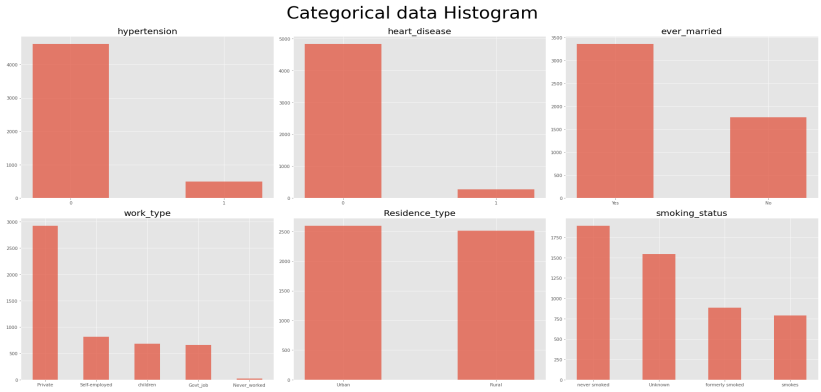
그와 달리 smoking_status의 라벨은 never smoked, Unknown, formerly smoked, smokes로 총 4가지 이다. 나쁨, 중간, 좋음과 같은 단계적 범주형데이터의 경우 1,2,3과 같이 증가하는 숫자로 인코딩할 수 있겠지만, smoking status의 경우 Unkown이 포함되어 있어 위의 예시처럼 인코딩하기는 어렵다고 판단하여 차후 categorical encoding시 이를 고려하여 진행하였다.

work type의 경우 라벨이 여러개인 점, 각 라벨 간의 관계를 대,소 상,하 관계로 보기 어려운 점을 고려하여 one-hot encoding을 진행하였다.

```
[ ]  plt.style.use("ggplot")

     # 히스토그램 을 사용해서 데이터의 분포를 살펴봅니다.
     plt.figure(figsize=(25,20))
     plt.suptitle("Categorical data Histogram", fontsize=40)

     # id는 제외하고 시각화합니다.
     cols = ['hypertension', 'heart_disease', 'ever_married', 'work_type',
             'Residence_type','smoking_status']
     for i in range(0, len(cols)):
         plt.subplot(3,3,i+1)
         plt.title(cols[i], fontsize=20)
         temp = df[cols[i]].value_counts()
         plt.bar(temp.keys(), temp.values, width=0.5, alpha=0.7)
         plt.xticks(temp.keys())
     plt.tight_layout(rect=[0, 0.03, 1, 0.95])
     plt.show()
```



3.2.4 범주형 변수의 stroke과의 상관관계

앞서 제시한 3가지 Research question중 나이, 평균 혈중 당수치, bmi와 stroke의 관련성을 파악하고자 2.3.2에서 boxplot을 통해 stroke의 여부에 따른 수치형 변수의 분포를 확인하였다. 성별, 질병 여부, 흡연 여부와 같은 범주형 변수와 stroke의 상관관계를 확인하기 위해서는 앞서 활용한 방법으로는 그 상관관계를 확인하기 어렵다. 이러한 범주형 데이터간의 상관관계는 Cramers'V 계수로 확인해볼 수 있다.

$$V = \sqrt{\frac{\chi^2}{n\,(q-1)}}$$

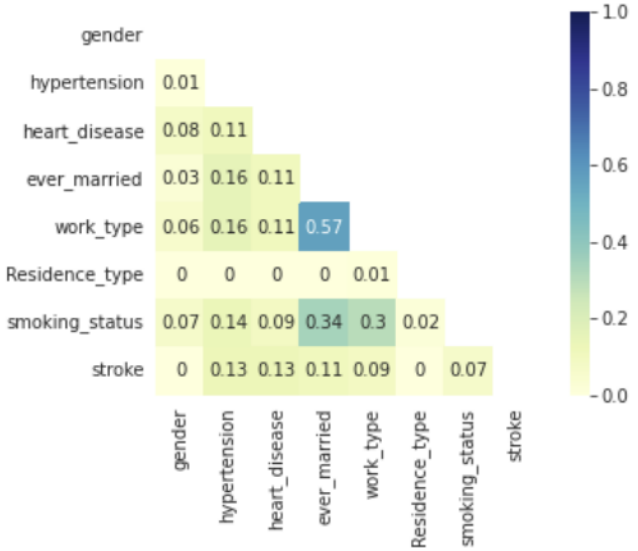
χ^2 : 카이자승 공식에 의해 구함

n : 총사례수

q : 줄 또는 칸의 유목수 중에 적은 숫자

```
[ ]  def cramers_V(var1,var2) :
       crosstab = np.array(pd.crosstab(var1, var2, rownames=None, colnames=None)) # Crosstab building
       chi2 = chi2_contingency(crosstab)[0]
       n = np.sum(crosstab)
       phi2 = chi2 / n
       r, k = crosstab.shape
       phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))
       rcorr = r - ((r-1)**2)/(n-1)
       kcorr = k - ((k-1)**2)/(n-1)
       return np.sqrt(phi2corr / min((kcorr-1), (rcorr-1)))
```

위와 같이 계산할 수 있는 Cramers'V 계수는 두개의 범주형 변수가 얼마나 강력하게 연관되는지를 표현한다.이 값은 항상 0~1사이의 양수값을 가지는데, 1에 가까울수록 그 연관관계가 크다고 해석할 수 있다. 대체로 계수 ≤ 0.2일때는 약한 상관관계,0.2 < 계수 ≤ 0.6는 적당한 상관관계, 계수 > 0.6의 경우 강한 상관관계를 가진다고 해석한다.



결과는 위의 히트맵과 같이 나타난다. work_type과 ever_married 컬럼의 상관관계, smoking status와 ever_married, work_type 외에는 모두 두 범주형 변수 간 약한 상관관계를 나타낸다. 하지만 이는 단순 두 개 변수만의 상관관계로, 여러개 변수가 합쳐진 상관관계 효과는 확인할 수 없다. 그렇기 때문에 해당 변수들을 현재 단계에서 모두 제거하기 보다는, 이후 전진선택법이나 후진소거법과 같은 변수 선택 단계를 거쳐 변수를 제거하거나 삭제하도록 한다.

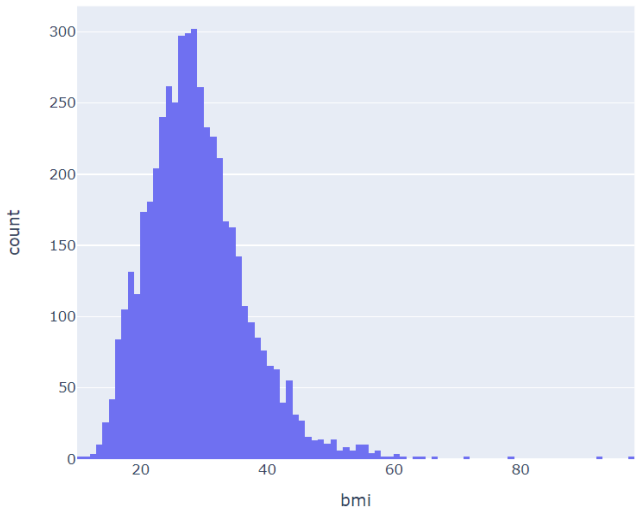
3.3 Imputation

Research question인 stroke의 예측 및 질병 분류 문제 등의 해결을 위해, 전처리 과정의 일환으로 결측치 처리는 필수적이다. 특히, 문제 해결 방법으로서 머신러닝 모델링을 활용할 경우, 결측치를 잘못된 숫자로 대체하거나 아예 제거할 경우, 해당 변수로 인해 모델이 데이터를 제대로 학습하지 못할 가능성이 크다. 따라서, 해당 데이터의 분포를 확인하거나, 시계열 데이터라면 앞,뒤 레코드의 평균값을 활용한다거나 하는 다양한 방법을 고려하여 결측치 처리를 진행해야한다.

전체 dataframe의 결측치를 다음의 방법으로 확인해본 결과 bmi컬럼에서만 총 201건 결측치를 발견하였다. 총 5110건의 레코드 중 201의 결측치는 4%에 해당하는 숫자로 제거하기 보다는 특정 통계량으로 대체하여 데이터의 수를 유지하는 것이 옳은 방향이라고 판단하였다. 어떤 통계량으로 결측치를 대체할지 결정하기 위해 bmi의 히스토그램을 참고하였다.

```
[ ]  df.isnull().sum()

gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```



EDA를 통해 확인한 바와 같이 대체로 정규분포를 따른다고 볼 수 있겠으나 70이상의 매우 큰 bmi의 레코드들을 몇 건 확인할 수 있었다. 이에 이상치에 덜 민감한 median으로 결측치를 채워주었다.

```
[ ]  med = np.median(np.array(df['bmi'].dropna()))
     med

     28.1

[ ]  df = df.fillna(med)
```

```
[ ]  df.isnull().sum()

gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

3.4 Data Sampling

분류 모델의 데이터 학습이 적절하게 이루어지도록 하려면, 분류 대상이 되는 target label의 수를 비슷하게 맞추어야한다. 분류 라벨이 한쪽으로 치우쳐져 있는경우, 예를 들어 99%의 0과 1%의 1인 라벨의 데이터의 경우 모든 데이터를 0으로 분류만 해도 99%의 정확도를 가지게 되고, 이는 1% 라벨 1을 분류하기 위한 모델의 목적을 달성했다고 보기 어렵다. 우리가 사용한 데이터셋은 의료 데이터로, 질환이 있는 환자가 일반인보다 적기 때문에 클래스의 불균형을 고질적인 문제로 가지고 있다. 데이터셋의 클래스의 불균형을 해소하여 모델의 목적을 달성하기 위해 Data Sampling 과정이 필요하다.

Random Undersampling을 사용하여 많은 레이블을 가진 데이터셋을 적은 레이블을 가진 데이터셋의 수준으로 감소시키고자 하였다. Sampling을 하기 전에 shuffling을 하여 row 순서를 섞어주었다.

```
x_shuffled = sklearn.utils.shuffle(x, random_state=2022)
y_shuffled = sklearn.utils.shuffle(y, random_state=2022)
```

```
x_us, y_us = RandomUnderSampler(random_state=2022).fit_resample(x_shuffled, y_shuffled)
```

범주형 변수를 가지는 컬럼의 인덱스 배열을 전달하여 data sampling 모델을 정의하고 이를 x와 y에 대해 학습시켜 x_us, y_us을 반환받는다.

3.5 Data Partitioning

모델이 데이터를 학습하는 것 뿐 아니라, 실제 데이터에 모델이 잘 예측하는지 확인이 필요하다. 이를 위해 train data와 test data를 분리하여 train data에서만 모델의 학습을 진행하고, 실제 데이터에 대한 예측 성능을 test data로 확인한다.

Data sampling 이후 row 수가 498개로, 유의미한 수의 test set을 확보하기 위하여, train:test=8:2의 비율로 데이터를 나누었다. 또한 타겟 변수인 'stroke'는 범주형 변수이기에 stratified sampling을 사용할 수 있는데, random sampling과 달리 데이터 비율을 반영할 수 있어 stratified sampling을 사용하였다.

```
X_train, X_test, y_train, y_test = train_test_split(x_us, y_us, test_size=0.2, stratify = y_us, random_state = 2022)
```

```
print('train set: %d, test set: %d' %(len(y_train), len(y_test)))
```

```
train set: 398, test set: 100
```

```
print('train set 1 비율: %f, 0 비율: %f' %(y_train.value_counts(1)[0], y_train.value_counts(1)[1]))
print('test set 1 비율: %f, 0 비율: %f' %(y_test.value_counts(1)[0], y_test.value_counts(1)[1]))
```

```
train set 1 비율: 0.500000, 0 비율: 0.500000
test set 1 비율: 0.500000, 0 비율: 0.500000
```

train set의 데이터 개수가 398개, test set의 데이터 개수가 100개로 8:2 비율로 data partition이 된 것을 확인할 수 있다. 또한, train set과 test set의 데이터 비율이 0.500000으로 같은 것을 확인할 수 있다.

3.5 Categorical encoding

모델 학습시 string data는 숫자형으로의 변환이 필요하다. 모델은 자연어 형식의 데이터를 학습할 수 없기 때문이다.

전체 데이터 중 'gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status' 5개의 변수는 문자열 값을 가진 범주형 변수였기 때문에 이에 대한 인코딩 과정이 필요했다.

	gender	ever_married	work_type	Residence_type	smoking_status
483	Male	Yes	Private	Rural	smokes
210	Female	Yes	Private	Rural	smokes
104	Male	No	Private	Urban	Unknown
474	Female	Yes	Private	Rural	never smoked
197	Male	Yes	Self-employed	Urban	smokes

그 중 2개의 값을 가지는 gender, ever_married, Residence_type 변수는 scikit-learn의 LabelEncoder() 함수로 변환하였다. 다음은 변환 후의 라벨 설명이다.

Gender	
Male	Female
1	0

Ever married	
No	Yes
0	1

Residence_type	
Rural	Urban
0	1

나머지 변수인 work_type과 smoking_status는 각각 "children", "Govt_job", "Never_worked", "Private", "Self-employed", 그리고 "formerly smoked", "never smoked", "smokes", "Unknown" 등 2개 이상의 값을 가지기 때문에 get_dummies 변수로 One-hot 인코딩을 실행하였다.

4. Decision Tree Modeling

Decision Tree는 간단히 일련의 필터과정으로 어떤 attribute를 사용하려 먼저 분리하였을때 더 좋은 결과를 낼 수 있는지 결정하는 알고리즘이라고 할 수 있다. 이때 어떤 특성을 먼저 사용할지의 기준으로 불순도(Impurtiy)를 사용하게 된다. 불순도는 한 그룹내에 얼마나 여러 라벨이 섞여있는지를 의미하는 말로, 불순도가 작을수록 한가지 라벨로 그룹이 구성되어있고, 불순도가 클 수록 여러 라벨이 섞여있음을 의미한다고 이해할 수 있다. 이러한 불순도를 측정하는 방법으로는 대표적으로 Entropy와 Gini index가 있다.

- Entropy

Entropy는 무질서한 정도를 정량화한 값으로 다음과 같이 계산할 수 있다.

$$E = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

여기서 pi는 각 노드에서의 전체 샘플에서 각 라벨의 비율이라고 할 수 있다. 이러한 엔트로피 값이 작을수록 분류가 잘 되었다고 볼 수 있다.

- Gini index

지니계수는 정답이 아닌 라벨이 뽑힐 확률로, 클래스들이 공평하게 섞여있을수록 그 값이 커진다.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

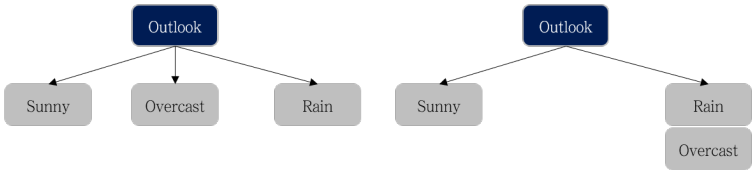
Information Gain은 부모 노드의 정보량(엔트로피 혹은 지니 계수)에서 자식 노드의 정보량을 뺀 것으로, 그 숫자가 클수록 정보의 순도가 더 높게 나뉘었다고 할 수 있다.

어떤 지표를 사용하느냐에 따라 크게 엔트로피 (Entropy)와 지니 계수 (Gini index) 기반의 알고리즘으로 나눌 수 있다.

1. Entropy기반 알고리즘
    - ID3
    - C4.5

2. Gini index기반 알고리즘
    - CART

ID3 CART



4.1 ID3

ID3는 가장 기본이 되는 알고리즘으로, C4.5, CART, CHAID 모두 ID3를 근간으로 한다. ID3는 엔트로피를 줄여나가는 방향, 즉 information gain이 커지는 방향으로 분기한다. 엔트로피는 범주형 변수에만 가능하기 때문에, 범주형 데이터에만 사용할 수 있는 알고리즘이다.

4.2 C4.5

C4.5는 ID3를 보완한 알고리즘이다. 먼저 엔트로피를 사용한 ID3의 information gain 지표를 정교화하여 더 정확하게 가지를 분할할 수 있도록 한다. 또한 범주형 변수만 다룰 수 있었던 ID3와 달리 C4.5는 연속형 데이터도 가능하고, information gain 수식에 가중치를 더해 결측치가 있는 데이터도 한번에 처리할 수 있다는 이점이 있다. 나무가 깊어질수록 오버피팅의 위험이 있는데, C4.5는 사후 가지치기 방법을 씀으로써 오버피팅 문제를 해결한다.

4.3 CART

CART는 Classification and Regression Tree의 약자로, 지니 계수를 쓰는 알고리즘이다. ID3와 유사하지만 가장 큰 차이점은 가지의 분기 방법인데, ID3는 클래스별로 가지가 뻗는 반면 CART는 하나의 클래스와 나머지의 방식으로 가지가 뻗어나간다. 또한 CART는 회귀가 가능한 모형으로, 회귀 트리일 때는 지니 계수가 아닌 실제값과 예측값의 차를 지표로 사용한다.

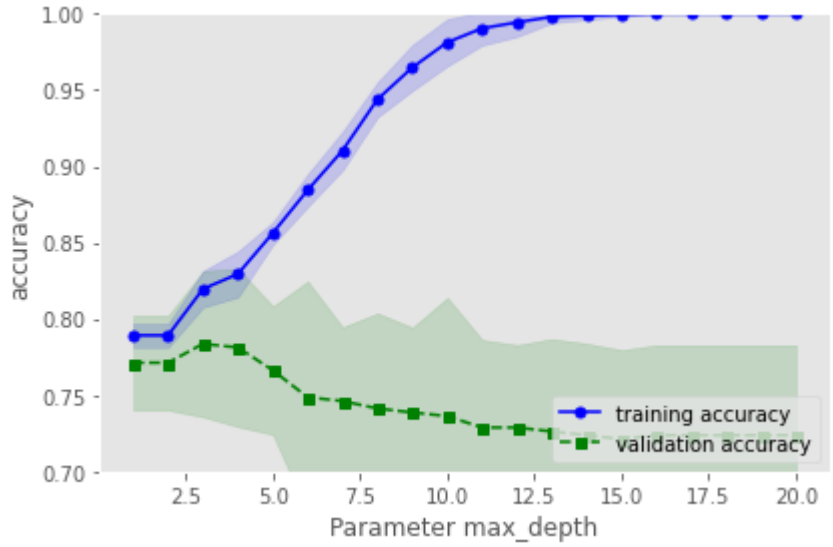
4.4 CHAID

CHAID는 Chi-Squared Automatic Interaction Detection의 약자로, CART와 유사하지만, 데이터를 분할하는 방식에서 차이가 존재한다. 먼저 CHAID는 오버피팅이 되기 전에 트리 형성을 멈춘다. 또한 CHAID는 최적의 분할을 찾기 위해 카이제곱 검정을 사용한다. 그렇기 때문에 범주형 자료에 대해서만 사용할 수 있고, 연속형 자료의 경우 F 검정을 분할의 지표로 사용한다.

4.5 Validation curve

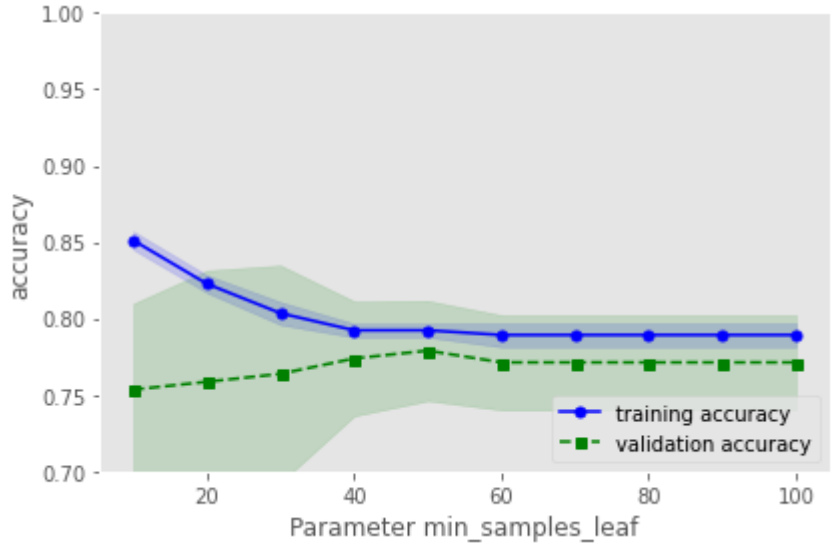
적절한 Decision Tree Classifier parameter를 찾기위해 대표적인 두가지 parameter인 Max_depth와 min_samples_leaf에대해 검정곡선을 그려보았다.

1. Max_depth



위 그림은 나머지 파라미터가 고정된 상태에서, max_depth의 변화에 따라 train, validation accuracy가 어떻게 변화하는지를 시각화 한것이다. Max_depth가 커질수록 training accuracy와 validation accuracy사이 gap이 커지는것을 확인할 수 있었다. 즉, max_depth가 지나치게 커질경우 오버피팅이 발생함을 확인할 수 있었다. 그중 Train accuracy와 validation accuracy가 모두 적절한 성능을 보이고 오버피팅이 발생하기 전인 max_depth = 3 선택하면 좋을 것이라 판단하였다.

2. min_samples_leaf



위 그림은 나머지 파라미터가 고정된 상태에서, min_samples_leaf의 변화에 따라 train, validation accuracy가 어떻게 변화하는지를 시각화 한것이다. min_samples_leaf가 커질수록 training accuracy와 validation accuracy사이 gap이 작아지는 것을 확인할 수 있었다. 즉, min_samples_leaf가 작을수록 오버피팅이 발생함을 확인할 수 있었다. 그중 Train accuracy와 validation accuracy가 모두 적절한 성능을 보이고 오버피팅이 발생하기 전인 min_samples_leaf = 50 선택하면 좋을 것이라 판단하였다.

4.6 Grid Search

다음으로는 Max_depth, min_samples_leaf 그리고 criterion으로서 Entropy와 gini계수 세가지를 종합적으로 고려하였을때 어떤 parameter에서 가장 최적의 결과를 얻을 수 있는지 grid search를 활용하여 알아보았다. 위에서 검정 곡선으로 선택한 결과와 비교했을때 Max_depth는 동일하게 3이 최적의 결과를 낸 반면 min_samples_leaf 는 10일때 그리고 gini계수를 criterion으로 사용했을 때 최적의 결과를 얻을 수 있었다. 위에서 설정한 최적의 경우와 다른 결과를 낸 이유로는 위에서는 한가지 파라미터의 변화만 확인한 반면 아래에서는 여러 파라미터를 종합적으로 고려했기 때문으로 꼽을 수 있다. 그때의 best Accuracy로는 0.791을 얻을 수 있었다.

```
param_range1 = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
param_range2 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
param_range3 = ['gini', 'entropy']

param_grid = [{'decisiontreeclassifier__max_depth': param_range1,
               'decisiontreeclassifier__min_samples_leaf': param_range2,
               'decisiontreeclassifier__criterion': param_range3}]

gs = GridSearchCV(estimator = pipe_tree,
                  param_grid = param_grid,
                  scoring = 'accuracy', #  Classification일때  'accuracy','f1', Regression 일때 'neg_mean_squared_error','r2' 등
                  cv=10,
                  n_jobs= -1) # 병렬 처리: -1은 전부를 의미

gs = gs.fit(X_train, y_train)

print(gs.best_score_)
print(gs.best_params_)
```

```
0.7914743589743589
{'decisiontreeclassifier__criterion': 'gini', 'decisiontreeclassifier__max_depth': 3, 'decisiontreeclassifier__min_samples_leaf': 10}
```

4.7 결과 해석

Stroke의 경우 질병의 조기 진단 및 예측이 매우 중요한 병이라고 할 수 있다. 의료 데이터 분석에서는 올바른 진단을 내리는 accuracy도 중요하지만, 질병을 제대로 질병이라고 판단하는 recall도 매우 중요하다. 재현율 recall을 확인해 보았을 때 0.84로 매우 높은 score를 가진다. 즉, 실제 병에 걸린 사람을 병에 걸렸다고 판단할 확률이 84%인 모델임을 알 수 있다.

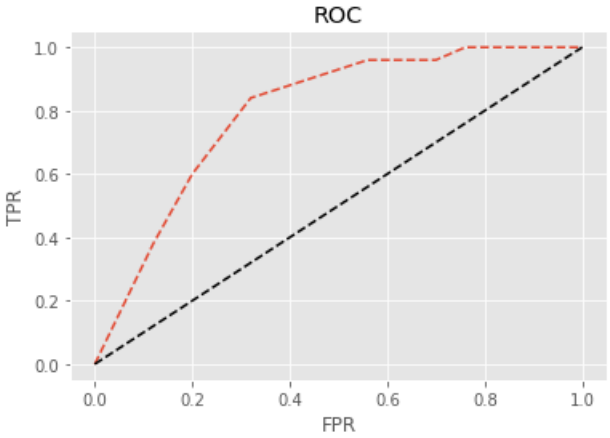
	Predict[0]	Predict[1]
True[0]	34	16
True[1]	8	42

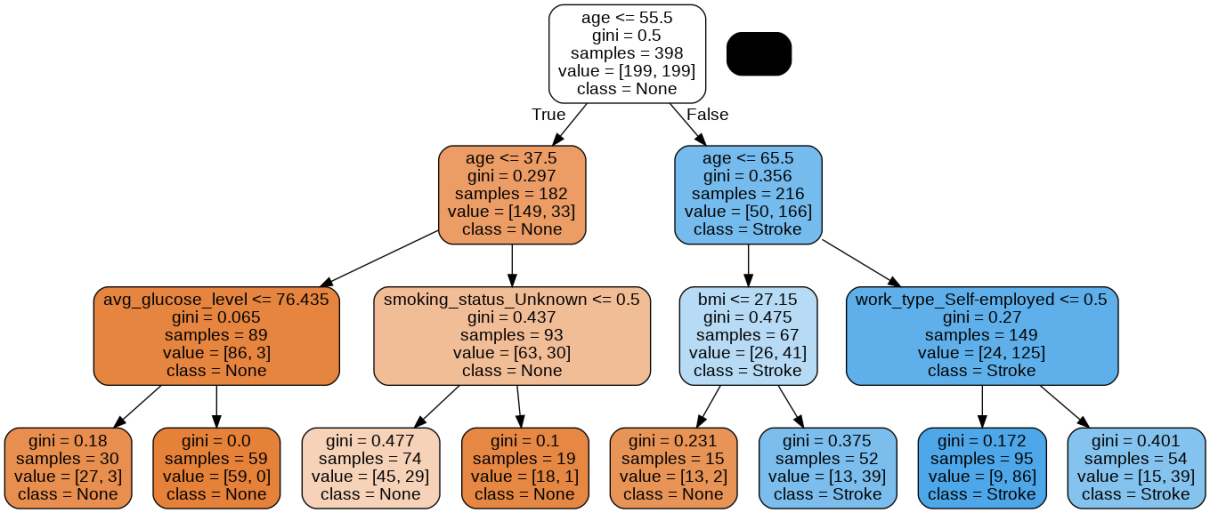
Classification Report

	precision	recall	f1-score	support
0	0.81	0.68	0.74	50
1	0.72	0.84	0.78	50
accuracy			0.76	100
macro avg	0.77	0.76	0.76	100
weighted avg	0.77	0.76	0.76	100

```
정확도 accuracy: 0.760
정밀도 precision: 0.724
재현율 recall: 0.840
F1-score: 0.778
AUC: 0.760
```

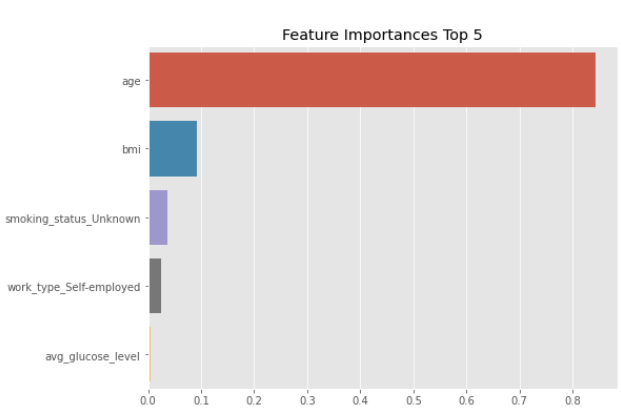
ROC curve를 시각화해 보았을때 곡선이 좌상단에 가까이 붙은 것을 확인할 수 있었고, 이는 모델이 잘 학습된 좋은 이진분류기임을 의미한다.





Classification 모델을 통해 뇌졸중을 진단받는 사람들의 특성에 대해서 설명할 수 있었다. Decision Tree의 가장 위쪽에 있는 규칙은 age<=55.5이다. 나이를 기준으로 분기한 결과 age가 55.5보다 크다고 분기된 leaf들에서 한 경우를 제외하고 뇌졸중을 진단받은 것을 확인할 수 있다. 따라서 나이라는 변수에 대한 중요도가 큰 것을 확인할 수 있다. 이후 진행된 bmi 규칙에 대하여 bmi가 27.15보다 작거나

같은 경우에 대해서는 나이가 많더라도 뇌졸중을 진단받지 않는다는 것을 알 수 있었다. bmi는 체질량 지수이다. 따라서 나이가 많더라도 체질량 지수가 적으면 뇌졸중 발병 확률이 낮아진다는 것을 의미한다. age가 55.5보다 작거나 같다고 분기된 leaf들에서는 뇌졸중을 진단받지 않을 확률이 크다는 경향을 보였다.




```
feature_importances.sort_values(ascending=False)

age                              0.842533
bmi                              0.092708
smoking_status_Unknown           0.036439
work_type_Self-employed          0.024156
avg_glucose_level                0.004165
gender                           0.000000
hypertension                     0.000000
heart_disease                    0.000000
ever_married                     0.000000
Residence_type                   0.000000
work_type_Govt_job               0.000000
work_type_Private                0.000000
work_type_children               0.000000
smoking_status_formerly smoked   0.000000
smoking_status_never smoked      0.000000
smoking_status_smokes            0.000000
dtype: float64
```

Decision Tree에서 상위에 속하는 변수일수록 의사결정에 중요한 영향을 미친다. 따라서 각 변수가 얼만큼의 중요성을 가지는지 확인하기 위해 Feature Importance를 확인해주었다.

Feature Importance 값을 확인한 결과 'age' 변수가 가장 중요한 의미를 가진다. 그 중요도가 0.842533으로 다른 변수들에 비해 중요하다는 것을 알 수 있다. 'age' 다음으로는 'bmi'가 0.092708의 중요도를 갖는 것을 알 수 있었다. 그러나 'age' 의 Feature Importance에 비해서 너무 작아 영향을 끼친다고 분석하기는 어렵다고 판단했다.

Decision Tree 시각화를 통해 확인한 결과와 동일하게 뇌졸중 예측에 가장 영향을 미치는 변수는 'age'인 것을 알 수 있었다.

II. Regression

1. Intro

대학원은 대학 학부 과정의 상위에 설치되어 있는 고도의 학술연구 기관 이자 연구 인력과 전문 직업 인력을 양성하는 교육기관으로, 학부 과정 이상의 고도의 학술연구를 희망하는 학생들이 추가적인 연구를 진행할 수 있다. 대학원 진학을 희망하는 이유는 다양하지만 최근 일자리를 얻지 못한 대학 졸업자들이 대학원 진학을 희망하는 경우가 증가했다. 한국교육개발원 교육 통계에 따르면, 2020년

국내 대학원 석사과정 입학생 수는 9만8402명으로 2019년(9만6694명)에 비해 1708명(1.8%) 늘었다. 학령 인구 감소로 대졸자는 2017년 68만5089명에서 2020년 66만6083명으로 줄었는데, 대학원 입학생 수는 오히려 늘어난 것이다.

대학원 진학률이 증가한 현상이 청년 실업과 관련 있다는 의견이 제시되었다. 민간 기업들의 신규 채용이 줄어든 것과 더불어 공기업 등 공공 부문도 비정규직을 정규직화하는 과정에서 청년 채용을 줄이면서 갈 곳을 잃은 대학 졸업자들이 발생한 것이다. 취업난을 겪고 있는 대학 졸업자들이 경력 공백을 만들지 않기 위해 대학원에 진학하는 경우도 발생했다.

이렇게 대학원 진학이 과열된 만큼, 대학원 합격에 대한 경쟁도 심화되었다. 하지만 경력 공백을 만들지 않기 위해 대학원 진학을 선택하더라도 대학원 진학이 청년 실업에 대한 완전한 해결책이 될 수는 없다. 이런 상황에서 만약 대학원에 합격할 확률을 예측할 수 있다면 합격할 확률이 낮은 경우 자신의 단점을 보완할 수 있고 앞으로의 진로에 대해서 다시 한 번 대학원을 고려해보는 계기가 될 수 있을 것이다. 이에 따라 본 실습 과정에서는 '대학원 합격 확률' 데이터를 활용하여 Decision Tree 알고리즘으로 Regression을 진행하고 대학원에 합격할 확률을 예측해보고자 한다.

2. Dataset

2.1 Data Explanation

본 실습에서 Decision Tree Regression과정을 진행하기 위해 '대학원 합격 확률' dataset을 이용하였다. 데이터를 선정한 기준은 다음과 같다.

1) 훈련세트(Feature나 관측치의 개수)가 너무 큰 것을 제외한다.
2) 특성 값 대부분이 0으로 구성된(희소 - sparse) Dataset은 제외한다.
3) Target이 연속형 변수이어야 한다.

훈련세트가 너무 큰 경우, Iteraion을 반복할 때 마다 distance metric을 업데이트 할 때 계산비용이 많이 듦으로 예측이 느려질 수 있다. 또한 특성 값 대부분이 0인 경우 모델이 잘 작동하지 않기 때문에 제외하였다.

9개의 변수는 모두 정수 혹은 실수로 구성되어있다.

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          400 non-null    int64
 2   TOEFL Score        400 non-null    int64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
```

Attribute Information

1. Serial No. : 학생 번호
2. GRE Scores (out of 340): GRE 점수 (340점 만점)
3. TOEFL Scores (out of 120): TOEFL 점수 (120점 만점)
4. University Rating (out of 5): 대학 평가 (5점 만점)
5. Statement of Purpose and Letter of Recommendation Strength (out of 5): 지원서와 추천서의 강도 (5점 만점)
6. Undergraduate GPA (out of 10): 학부 GPA (10점 만점)
7. Research Experience (either 0 or 1): 연구 경험 여부 (0 혹은 1)
8. Chance of Admit (ranging from 0 to 1): 합격 여부 (0~1 사이의 확률)

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

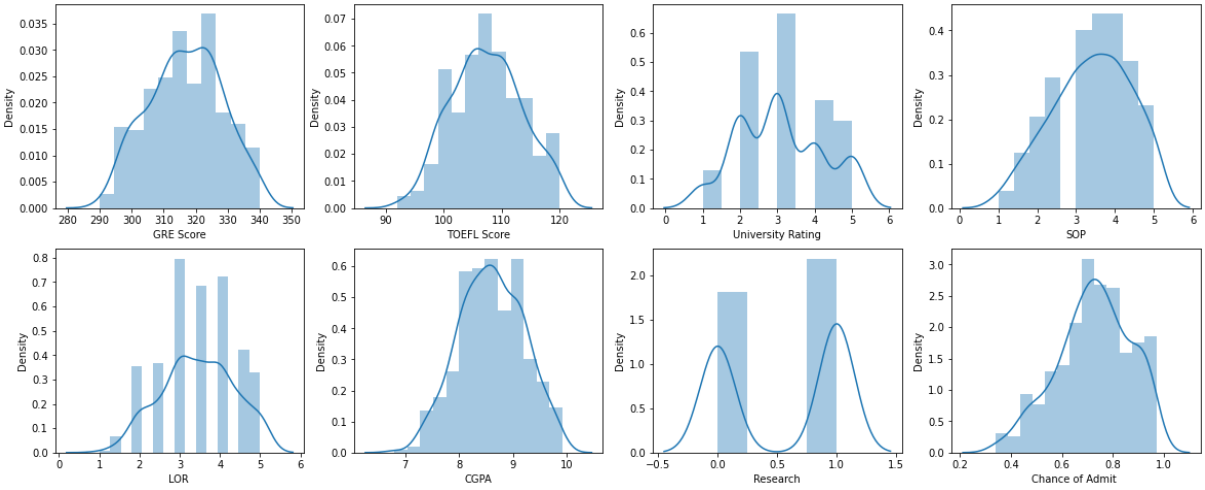
현재 데이터 셋은 모든 변수가 연속형 변수이다. 연속형 변수의 평균, 표준편차, 최솟값, 사분위수, 최댓값 등의 통계량을 확인했다.

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

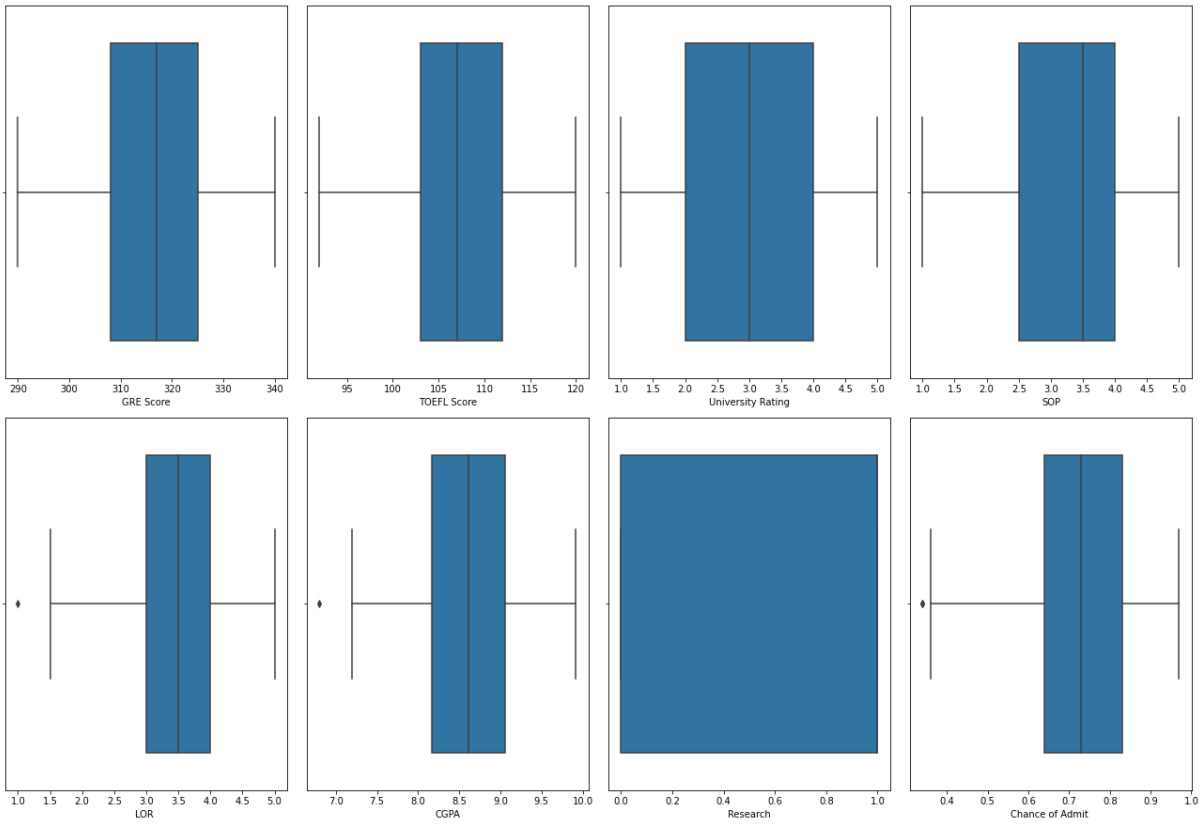
해당 데이터셋은 모든 데이터의 count가 400으로 동일한 결측치가 존재하지 않는 데이터인 것을 확인할 수 있었다.

2.2 EDA

Serial No. 는 학생 고유 번호로, KNN에 불필요한 변수이기 때문에 제외하고, 나머지 변수에 대해 연속형 변수의 분포를 확인하기 위해 각 변수 별 히스토그램을 나타내었다.



8가지 연속형 변수의 분포를 확인한 결과 변수 간 스케일이 다른 것을 시각적으로 확인할 수 있었다. 본 실습과정의 목표인 KNN 분류, 회귀과정에서 변수들의 스케일이 다를 경우 작은 단위 Feature에 영향을 많이 받게 된다. 이로 부터 오류가 발생할 수 있기 때문에 변수 간 스케일을 맞춰주기 위해 Scaling과정이 필요하다고 생각하였다.



연속형 변수의 분포를 boxplot을 이용해서 나타낸 결과, 이상치가 적은 데이터임을 확인할 수 있었다. 따라서 Standard scaler를 사용하여 Scaling을 진행해 데이터의 scale을 맞춰주었다.

2.2 Preprocessing

```
df.isnull().sum()

Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

Regression에서 사용하는 dataset의 결측치가 없는 것을 알 수 있다. 따라서 별도의 결측치 처리 없이 진행해주었다.

2.3 Sampling

```
[ ]  1 from sklearn.model_selection import train_test_split
     2 x_train, x_test, y_train, y_test = train_test_split(scaled_x, y_admit2, test_size = 0.2, random_state = 312)
     3 x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size = 0.2, random_state = 312)
```

```
1 print("train set:",x_train.shape)
2 print("valid set:",x_valid.shape)
3 print("test set:",x_test.shape)
```

```
train set: (256, 7)
valid set: (64, 7)
test set: (80, 7)
```

Dataset을 train:test=8:2의 비율로 나누었으며, train을 다시 train : valid = 8:2의 비율로 나누어 모델의 성능을 평가할 때 사용하였다. Train/Valid/Test로 구분한 결과 각각 256,64,80개의 데이터가 존재하는 것을 확인할 수 있었다.

2.4 Outlier Detection

```
def outlier_iqr(data, y, column):
    fraud = data[column]
    global lower, upper
    q25, q75 = np.quantile(data[column], 0.25), np.quantile(data[column], 0.75)
    iqr = q75 - q25
    # outlier cutoff
    cut_off = iqr * 1.5
    # lower와 upper bound 값 구하기
    lower, upper = q25 - cut_off, q75 + cut_off

    data1 = data[data[column] > upper]
    data2 = data[data[column] < lower]

    outlier_index = fraud[(fraud < lower) | (fraud > upper)].index
    data.drop(outlier_index, axis=0, inplace = True)
    y.drop(outlier_index, axis=0, inplace = True)
    # 이상치 총 개수 구하기
    print(i, '의 총 이상치 개수는', data1.shape[0] + data2.shape[0], '이다.')

    return data, y
```

```
for i in x_train.columns :
    outlier_iqr(x_train, y_train, i)
```

```
GRE Score 의 총 이상치 개수는 0 이다.
TOEFL Score 의 총 이상치 개수는 0 이다.
University Rating 의 총 이상치 개수는 0 이다.
SOP 의 총 이상치 개수는 0 이다.
LOR  의 총 이상치 개수는 1 이다.
CGPA 의 총 이상치 개수는 0 이다.
Research 의 총 이상치 개수는 0 이다.
```

Outlier Detection을 진행한 결과, 'LOR' 컬럼에서만 1개의 이상치가 관측되었고 다른 컬럼에서는 발견되지 않았다. 따라서 이상치의 영향이 크지 않다고 판단하여 이상치 제거 과정을 별도로 진행하지 않았다.

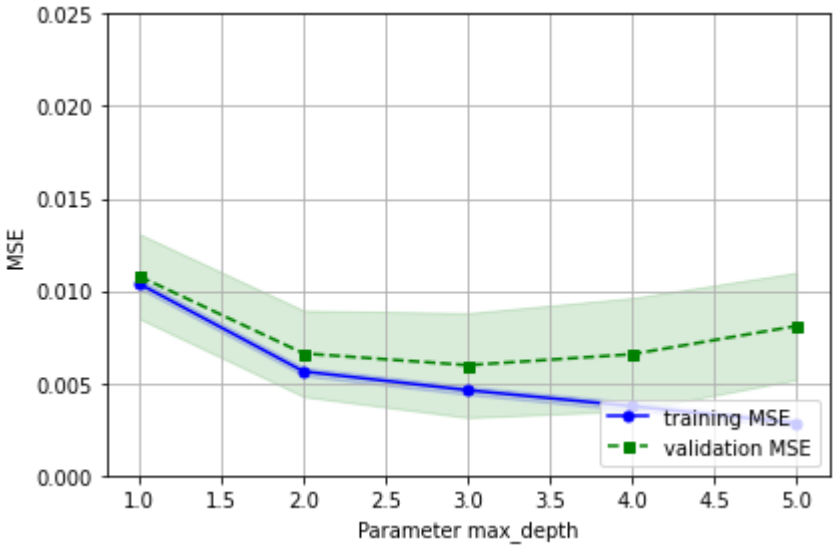
3. Regression Decision Tree

앞선 전처리 과정을 바탕으로 Decision Tree 알고리즘을 이용해 Regression을 진행해보았다.

3.1 Validation curve

과대적합 문제를 확인하기 위해 검정곡선을 그리고 Regression Decision Tree를 만들기 위해서는 max_depth의 validation curve를 그려 적절한 값을 파악해주었다.

3.2.1 Max depth



Max_depth값의 변화에 따라 그래프를 그려본 결과 위와 같았다. MSE가 작게 나타나는 0 ~ 0.025 구간에 대하여 그래프를 다시 그려본 결과 validation MSE가 가장 작게 나타난 지점이 max_depth가 3일때인 것을 알 수 있었다. 또한 max_depth가 3이상인 지점에서 training과 validation의 편차가 커지고 있는 것이 확인되었다. 따라서 최적의 max_depth는 train MSE와 validation MSE가 모두 적절한 성능을 내며 둘 사이의 편차가 커지지 않아 오버피팅이 발생하기 전인 3이라고 판단하였다.

3.2.2 Hyper Parameter Tuning

```
print(-gs.best_score_)
print(gs.best_params_)
```

```
0.005999472220727556
{'decisiontreeregressor__criterion': 'mse', 'decisiontreeregressor__max_depth': 3, 'decisiontreeregressor__min_samples_leaf': 10}
```

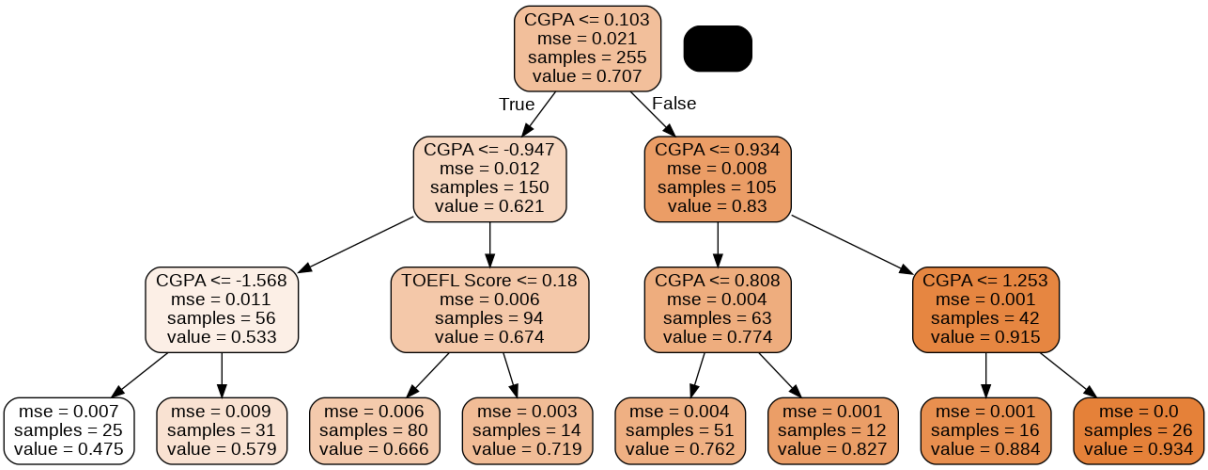
GridSearchCV를 통해 Hyper Parameter Tuning을 진행하여 best MSE score를 확인한 결과 0.006정도 인것을 확인할 수 있었다. 또한 MSE 값을 기준으로 최적의 max_depth값과 min_sample_leaf를 확인한 결과 다음과 같았다.

1) max_depth = 3
2) min_sample_leaf = 10

따라서 최적의 파라미터로 모델을 생성하고 모델을 이용해 target 값을 예측해주었다.

3.2.3 Regression Decision Tree

Regression Decision Tree를 시각화한 결과 위와 같았다. Max depth를 10로 설정하였기 때문에 한눈에 알아보기 쉬운 규칙을 찾아보기에는 힘들 것으로 보인다. 따라서 기대수명이 가장 높게 나오는 규칙과 가장 낮게 나오는 규칙을 따라가보고 기대수명을 높이기 위해서는 어떠한 변수들이 중요시되는지 파악해보려고 한다



Regression Decision Tree 모델을 시각화한 결과 위와 같았다. 가장 높은 합격확률을 갖기 위해서는 어떠한 변수들이 중요한지 그래프를 통해 알아보고자 한다. 따라서 가장 높은 대학원 합격확률을 갖는 leaf는 어떤 노드를 거치는지 확인해보았다.

- CGPA <= 0.103 (False)

- CGPA <= 0.934 (False)

- CGPA <= 1.253 (False)

'CGPA' 값에 따른 분류가 진행되고 있는 것을 확인할 수 있다. 또한 해당 노드의 mse는 'CGPA'는 학부 GPA로, 10점 만점으로 환산하여 나타낸 것이다. Scaling에 의해 -3.02 ~ 2.218 사이로 범위가 변경되었다. 따라서 CGPA가 높다는 것은 학부 GPA 즉, 학점이 높다는 것을 의미한다.

가장 낮은 대학원 합격확률을 갖는 leaf는 어떤 노드를 거치는지 확인해보았다.

- CGPA <= 0.103 (True)

- CGPA <= -0.947 (True)

- CGPA <= -1.568 (True)

‘CGPA’ 값에 따른 분류가 진행되고 있는 것을 확인할 수 있다. CGPA가 낮다는 것은 학부 GPA 즉, 학점이 낮다는 것을 의미한다. Decision Tree의 분기가 CGPA에 의해 진행되는 것을 통해 Regression 모델의 가장 중요한 변수는 CGPA인 것을 알 수 있다.

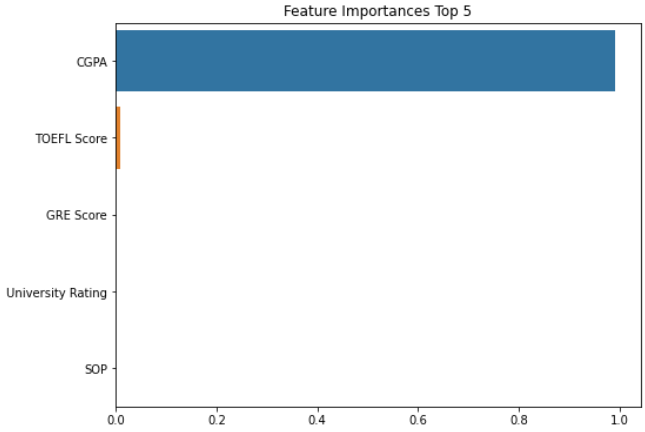
Regression Decision Tree 모델의 성능을 평가하기 위해 R2 score 와 MSE(Mean Squared Error)를 확인해보았다.

```
from sklearn.metrics import r2_score, mean_squared_error
print('R squared: %.3f' % r2_score(y_test, y_pred))
print('MSE: %.3f' % mean_squared_error(y_test,y_pred))
```

```
R squared: 0.745
MSE: 0.005
```

R2 score가 0.745, MSE가 0.005인 것을 확인할 수 있다.

3.2.4 Feature Importance




```
feature_importances.sort_values(ascending=False)
```

```
CGPA                0.991932
TOEFL Score         0.008068
GRE Score           0.000000
University Rating   0.000000
SOP                 0.000000
LOR                 0.000000
Research            0.000000
dtype: float64
```

Decision Tree에서 상위에 속하는 변수일수록 의사결정에 중요한 영향을 미친다. 따라서 각 변수가 얼만큼의 중요성을 가지는지 확인하기 위해 Feature Importance를 확인해주었다.

Feature Importance 값을 확인한 결과 ‘CGPA’ 변수가 가장 중요한 의미를 가진다. 그 중요도가 0.991932로 다른 변수들에 비해 중요하다는 것을 알 수 있다. ‘CGPA’ 다음으로는 ‘TOEFL Score’가

0.008068의 중요도를 갖는 것을 알 수 있었다. 그러나 'CGPA' 의 Feature Importance에 비해서 너무 작아 영향을 끼친다고 분석할 수 없었다.

Decision Tree 시각화를 통해 확인한 결과와 동일하게 대학원 합격확률 예측에 가장 영향을 미치는 변수는 'CGPA'인 것을 알 수 있었다.