07. Decision Tree

CART(Classification and Regression Tree)

1. Introduction

- **ID3, CART, CAHID, C4.5 알고리즘**

■ ID3

● 알고리즘

1. 전체 데이터를 포함하는 루트 노드를 생성한다.
2. 만약 샘플들이 모두 같은 클래스라면, 노드는 잎이 되고, 해당 클래스로 레이블을 부여한다.
3. 그렇지 않으면 정보이득이 높은(즉, 데이터를 가장 잘 구분할 수 있는) 속성을 선택한다. (이때 정보이득(불순도)은 **엔트로피의 변화**를 가지고 계산한다.)
4. 선택된 속성으로 가지(Branch)를 뻗어 하위 노드들을 생성한다.
   (각 하위 노드들은 가지의 조건을 만족하는 레코드들이다.)
5. 각 노드에 대해여 2 단계로 이동한다.

● ID3 특징

- 범주형 속성에 대해서만 사용이 가능하다. (엔트로피 기반 척도 수식이 범주형 속성에 대해서만 적용이 가능하기 때문에 그렇다.)
- 상위 노드에서 사용된 속성은 다시 사용하지 않는다.

■ CART

● 알고리즘

1. 전체 데이터를 포함하는 루트 노드를 생성한다.
2. 만약 샘플들이 모두 같은 클래스라면, 노드는 잎이 되고, 해당 클래스로 레이블을 부여한다.
3. 그렇지 않으면 정보이득이 높은(즉, 데이터를 가장 잘 구분할 수 있는) 속성을 선택한다. (이때 정보이득(불순도)는 **Gini Index(범주형), 분산의 차이(연속형)**를 가지고 계산한다.)
4. 선택된 속성으로 가지(Branch)를 뻗어 하위 노드들을 생성한다.
   (각 하위 노드들은 가지의 조건을 만족하는 레코드들이다.)

5. 각 노드에 대해여 2단계로 이동한다.

- **CART 특징**

- 후보 나무들을 여러 개 생성하고 그 중에서 최적의 나무를 찾아내는 방법을 사용

- CART 알고리즘은 Binary Split으로 가지를 뻗어나간다.

■ **CHAID**

- **알고리즘**

1. 전체 자료를 둘 이상의 하위노드(child node)로 반복적으로 분할한다.
2. 이 과정에서 설명변수의 범주의 쌍에 대한 반응변수의 유의한 차이가 없으면 설명변수의 범주들을 병합하며, 유의적이지 않은 쌍들이 없을 때까지 과정을 계속한다.
3. 각 설명변수에 대한 최고의 분할을 찾고, 모든 설명변수에 대한 유의성을 조사하여 가장 유의적인 설명변수를 선택한다.
4. 선택된 설명변수의 범주들의 그룹을 사용해 자료를 상호 배반인 부분집합으로 분할하며 각 부분집합에서 정지규칙중의 하나가 만족될 때까지 이 과정을 독립적으로 순환, 반복한다.

- **CHAID 특징**

- CHAID 알고리즘은 카이제곱 통계량(이산형 목표변수) 또는 F-검정(연속형 종속변수)을 이용해 다지 분리를 수행한다.

- 데이터를 Overfitting 하기 전에 나무 형성을 멈춘다.

■ **C4.5**

- **알고리즘**

1. 전체 데이터를 포함하는 루트 노드를 생성한다.
2. 만약 샘플들이 모두 같은 클래스라면, 노드는 잎이 되고, 해당 클래스로 레이블을 부여한다.
3. 그렇지 않으면 정보이득이 높은(즉, 데이터를 가장 잘 구분할 수 있는) 속성을 선택한다. (이때 정보이득(불순도)은 Information gain ratio 를 가지고 계산한다.)

4. 선택된 속성으로 가지(Branch)를 뻗어 하위 노드들을 생성한다.
   (각 하위 노드들은 가지의 조건을 만족하는 레코드들이다.)
5. 각 노드에 대해여 2 단계로 이동한다.

- **CHAID 특징**

- C4.5는 ID3 알고리즘과 크게 다르지 않다. ID3의 단점들을 보완한 알고리즘이다.

> 정교한 불순도 지표 활용 / 연속형 변수 사용 가능 / 결측치가 포함된 데이터도 사용 가능 / 과적합을 방지하기 위한 가지치기로 개선이 됨

■ **각 알고리즘 비교**

알고리즘	평가지수	비고
ID3	**Entropy**	**다지분리(범주)**
CART	**Gini Index(범주), 분산의 차이(수치**	**통계적 접근, Binary 분리**
CHAID	**카이제곱(범주), F검정(연속)**	**통계적 접근**
C4.5	**Information Gain Ratio**	**다지분리(범주) Binary 분리(연속)**

- **Classfication Tree**

의사결정나무(Decision Tree)을 만들 때 분류하고자 하는 데이터의 종속변수가 이산형 변수일 때 Classification Decision Tree를 만든다. Classification DT의 경우 가지를 나누는 분류 기준으로 Gini Entropy나 Chi-Square statistic을 활용하며, 이러한 기준을 따라 만든 DT의 적합성은 misclassification rate를 통해 판단한다.

■ Gini Index

Gini index란, 전체 데이터 중 특정 범주에 속하는 데이터의 비율을 모두 제외한 값으로, 비율을 제곱해서 뺀다는 특징을 가진다. 다음과 같은 계산식을 통해 Gini index를 구할 수 있다.

$$G = 1 - \sum_{j=1}^{c} \left(\frac{n_j}{n}\right)^2$$

위의 식에서 n은 현재 노드에 있는 데이터의 수를 말하며, n_j는 현재 노드에서 j번째 범주에 속하는 데이터의 수, c는 범주의 개수를 의미한다. 다양성이 높을수록 Gini 계수의

값이 크고, 다양성이 낮고 비슷할수록 Gini 계수의 값이 작다.

데이터를 나누기 전 상태에서의 Gini 계수와 데이터를 기준에 따라 나눈 후의 각 노드의 Gini 계수를 비율에 맞게 더한 값의 차이에 따라 해당 분류 기준을 선정할지 다른 분류 기준을 선정할지를 결정한다. 차이에 대한 식은 다음과 같다.

$$\Delta G = G_0 - \{\left(\frac{n_L}{N}\right) G_{Left\ node} + \left(\frac{n_R}{N}\right) G_{Right\ node}\}$$

G0는 나누기 전 상태에서의 Gini 계수를 의미하며, 각 노드 앞의 분수는 전체 데이터에서 각 노드에 해당하는 데이터의 개수를 의미한다. ΔG값이 클수록 해당 변수가 분류에 적합하다고 판단할 수 있다.

- Entropy

엔트로피는 불순도를 의미하며, 식은 다음과 같다.

$$Entropy = -\frac{N_{class1}}{N_{total}} log_2\left(\frac{N_{class1}}{N_{total}}\right) - \cdots - \frac{N_{classl}}{N_{total}} log_2\left(\frac{N_{classl}}{N_{total}}\right)$$

$$= -P_i \sum_{i=1}^{l} log_2 P_i$$

Pi는 특정 영역에 속하는 데이터 중, i번째 범주에 속하는 데이터의 비율을 의미한다. 엔트로피를 최소화하는 방향, 즉 분류 전 엔트로피와 분류 후 엔트로피의 차이를 가장 크게 만드는 방향으로 분류를 진행한다. 이러한 엔트로피 차이를 information gap이라고 하며 이 값이 가장 크도록 분류를 진행한다.

- Chi-Square statistic

가지를 나누는 분류 기준으로 Chi-Square test를 활용할 수 있는데, 이때 Chi-square 통계량을 분리기준으로 설정할 수 있다. Chi-square의 통계량 식은 다음과 같다.

$$x^2 = \sum_{i,j} \frac{(f_{i,j} - E_{i,j})^2}{E_{i,j}}$$

Eij는 분포의 동질성을 의미하며, 다음과 같이 계산한다.

$$E_{i,j} = \frac{(f_i \times f_j)}{f_{..}}$$

카이제곱 통계량이 작다는 것은 p-value가 크다는 것을 의미하며 설명변수의 각 범주에 따른 종속변수의 분포가 동질적임을 의미한다. 따라서 카이제곱 통계량을 분류기준으로 선정하면 p-value의 값이 작을 때를 선택해 분류한다.

- **Regression Tree**

Regression Tree는 Classfication Tree와 같이 트리기반으로 한다. 즉, 회귀를 위한 트리를 생성하고 이를 기반으로 회귀 예측을 하는 것 이다. Regression Tree는 Classification Tree와 크게 다르지 않다. 다만 leaf node에서 예측 결정값을 만드는 과정에 차이가 있다. Classification Tree는 특정 클래스 레이블을 결정하지만, Regression Tree는 리프 노드에 속한 데이터 값의 평균 값을 구해 회귀 예측값을 계산한다. 가지치기(정보량, 불순도) 계산으로는 다음과 같은 방법이 있다.

- 분산 감소

CART 에서 소개된 분산 감소 기법은 목표 변수가 연속적일 경우(회귀 트리)에서 자주 사용된다. 즉, 만약 다른 방법에서 분산 감소 기법을 사용하기 위해서는 목표 변수를 적용하기 이전에 먼저 이산화 과정을 거쳐야함을 의미한다. 노드 N 에 대한 분산 감소는 노드의 분할로 인해 발생하는 목표 변수 x 의 분산의 총 감소로 정의된다.S,S_t, 그리고 S_f 를 각각 분할 전 샘플 지표의 설정, 분할 테스트가 참일 때에 해당하는 샘플 인덱스, 분할 테스트가 거짓일 때에 해당하는 샘플 인덱스라 할 때, 다음과 같이 나타난다.

$$I_V(N) = \frac{1}{|S|}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right)$$

- MSE

실제 값과 예측 값의 차이를 제곱하여 평균화해 계산하는 방법이다. 다음과 같이 표현이 된다.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

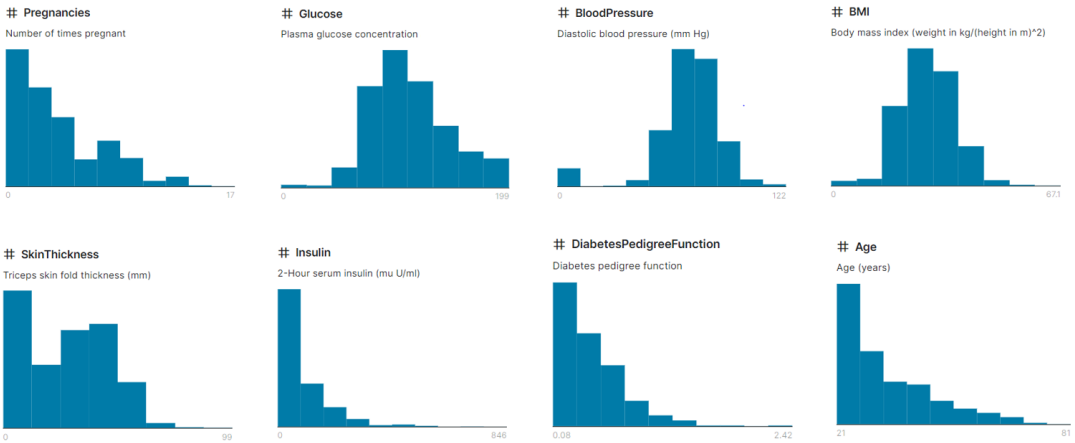
2. Classifcation 실습

2.1 데이터 설명 및 전처리

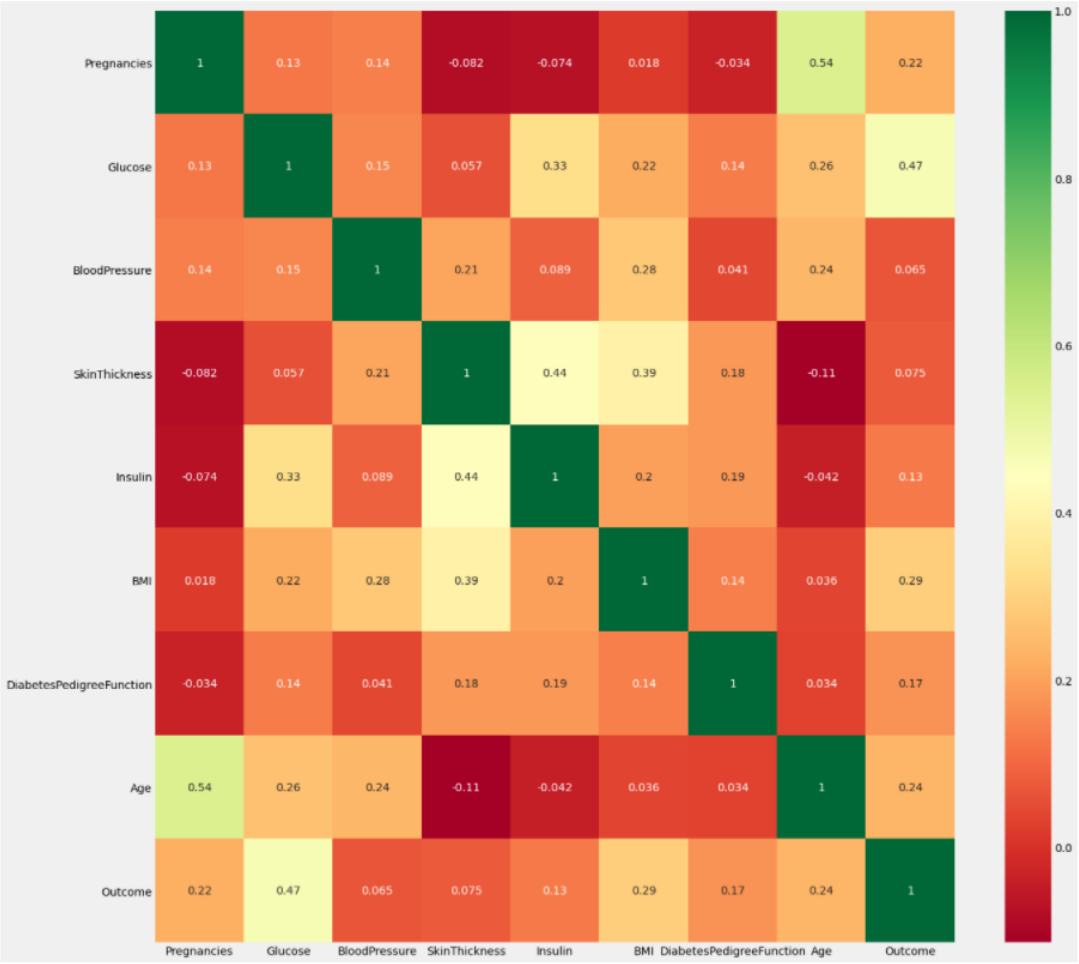
이번 실습에 활용할 데이터로 지난주 활용한 캐글의 당뇨병 진단 결과 데이터를 선택하였다. 해당 데이터는 미 국립 당뇨, 소화기병 및 신장병 연구소가 발표한 데이터로 총 768명의 21살 이상 여성 북미 원주민의 후손들의 나이, 인슐린 레벨, BMI, 등의 독립변수와 그들의 당뇨 진단 결과를 포함하고 있다.

총 8개의 독립변수와 1개의 종속변수로 구성되어 있으며 독립변수와 각 변수의 데이터가 가진 특징, 데이터 분포는 다음과 같다.

독립변수	평균	표준편차	최소값	최댓값
임신 횟수	3.85회	3.37	0	17
혈중 포도당 농도	12mg/dl	32	0	199
이완기 혈압	69.1mmHg	19.3	0	122
삼두근 피부 주름 두께	20.5mm	15.9	0	99
혈중 인슐린 농도	78.9uU/ml	115	0	128
BMI 지수	32	7.88	0	67.1
가족력에 의한 당뇨 가능성 지표	0.47	0.33	0.08	2.42
나이	33.2	11.8	21	81



독립변수들은 모두 연속형 변수였으며 임신 횟수, 혈중 인슐린 농도, 가족력에 의한 당뇨 가능성 지표, 나이는 좌측으로 치우친 분포를 띠는 것을 그래프를 통해 확인할 수 있었다. 그에 반해 혈중 포도당 농도, 이완기 혈압, BMI 지수는 비교적 정규분포에 가까운 분포를 띠는 것을 확인할 수 있었다.



독립변수들 간의 상관관계가 존재하는지 파악하기 위해 차트를 그려보았고, 차트는 위와 같이 나타났다. 매우 높은 상관관계를 가지는 독립변수는 없는 것으로 파악되었고, 나이와 임신 횟수, 혈중 인슐린 농도와 삼두근 피부 주름 두께정도만이 0.4정도의 양의 상관관계를 갖는다는 것을 확인할 수 있었다.

종속변수는 당뇨병을 진단받았는지를 나타내는 이항변수로, 당뇨병을 진단받은 경우 1, 그렇지 않은 경우 0으로 표시했다. 768명 중 268명이 당뇨병 진단을 받았고, 500명은 당뇨병이 없다고 진단받았다.

분석을 위해 데이터의 전처리과정을 진행하였다.

먼저 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI' 데이터가 0이 나오는 것은 상식에 맞지않으므로 평균으로 대체 해주는 과정을 거쳤다. 다음은 대체후 데이터를 나타낸다.

```
diabetes_data['Glucose'].fillna(diabetes_data['Glucose'].mean(), inplace = True)
diabetes_data['BloodPressure'].fillna(diabetes_data['BloodPressure'].mean(), inplace = True)
diabetes_data['SkinThickness'].fillna(diabetes_data['SkinThickness'].median(), inplace = True)
diabetes_data['Insulin'].fillna(diabetes_data['Insulin'].median(), inplace = True)
diabetes_data['BMI'].fillna(diabetes_data['BMI'].median(), inplace = True)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.686763	72.405184	29.108073	140.671875	32.455208	0.471876	33.240885	0.348958
std	3.369578	30.435949	12.096346	8.791221	86.383060	6.875177	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.202592	29.000000	125.000000	32.300000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

그뒤 전체 데이터를 train data와 test data로 나누고, 종속변수 0과 1의 개수가 같도록 데이터를 샘플링 하였다.

```
# train: test = 8:2 분리
from sklearn.model_selection import train_test_split
train_features, test_features , train_target, test_target = train_test_split(
    d_features, d_target, test_size = 0.2, random_state = 2021, stratify=d_target)
```

```
print(len(train_features))
print(len(train_target))

print(len(test_features))
print(len(test_target))
```

```
614
614
154
154
```

```
# Under Sampling: Y값을 각각 {0, 1} 214명씩
## sampling하기 전에 shuffling을 해주기(행 순서 섞기)
import sklearn
x_shuffled = sklearn.utils.shuffle(train_features, random_state=2021)
y_shuffled =sklearn.utils.shuffle(train_target, random_state=2021)

import imblearn
from imblearn.under_sampling import RandomUnderSampler
train_features_us, train_target_us = RandomUnderSampler(random_state=2021).fit_resample(x_shuffled, y_shuffled)
```

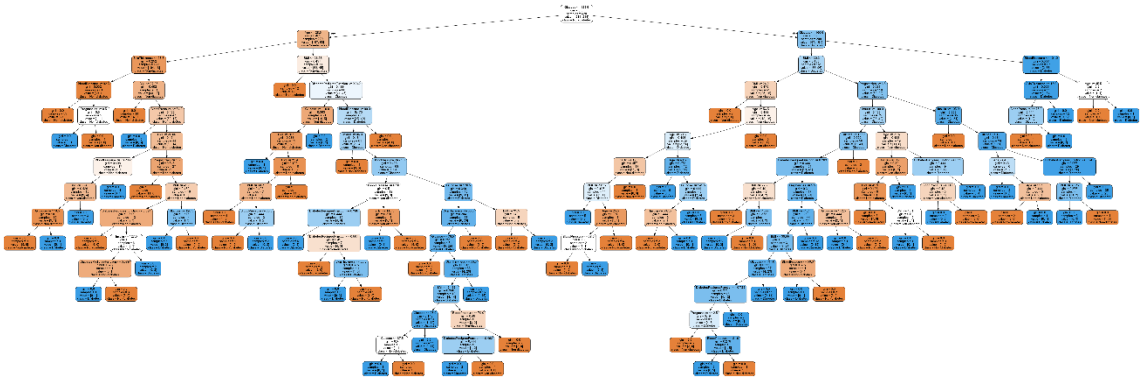
```
pd.DataFrame(train_target_us)['Outcome'].value_counts() # Y열의 데이터 개수 확인: total 428명
```

```
1    214
0    214
```

그 결과 종속변수가 1, 즉 당뇨병 진단을 받은 환자 214명과 정상 진단을 받은 214명의 데이터, 총 428명의 데이터를 얻었다. 이 데이터를 토대로 DT 모델링을 진행했다.

2.2 Gini index 기준 결정나무 적용

데이터를 구분한 이후 파이썬의 DecisionTreeClassifier를 통해 의사결정나무 모델을 적용하였다. 그리고 10-fold의 Cross Validation을 진행하여 각 경우에서 test data의 f1 score값을 구하였다. 다음은 Gini계수 기준 의사결정나무와 10-folds경우 각각의 f1값들이다.



```
from sklearn.model_selection import cross_validate

scores = cross_validate(estimator = tree,
                        X=train_features_us,
                        y=train_target_us,
                        scoring = ['f1'],
                        cv=10,
                        n_jobs= -1,
                        return_train_score=False)

print('CV f1: %s' % scores['test_f1'])
print('CV f1(Mean): %.3f (std: %.3f)' % (np.mean(scores['test_f1']),
                                         np.std(scores['test_f1'])))
```

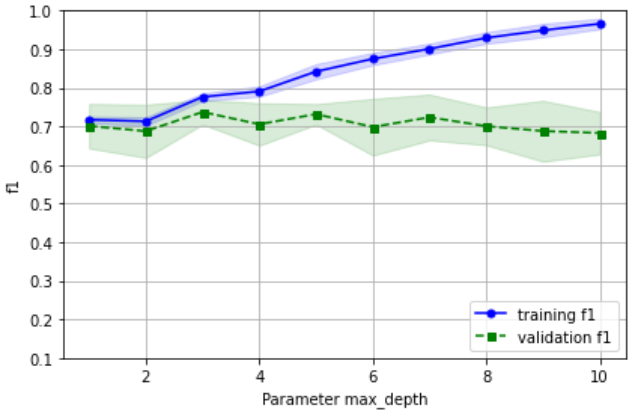
```
CV f1: [0.59090909 0.61904762 0.66666667 0.63414634 0.76        0.72340426
 0.55        0.75555556 0.68085106 0.74418605]
CV f1(Mean): 0.672 (std: 0.070)
```

해당 데이터는 당뇨병 여부 분류에 관한 모델이므로, 당뇨병을 가진 사람이 당뇨병이 없다고 오판하는 경우(False Negative)가 최대한 적어야 하며 당뇨병을 가졌다고 분류한 사람들 중 실제로 당뇨병을 가진 사람이 최대한 많아야 한다(True Positive).

즉 실제로 당뇨병이 있는 사람들 중 모델이 분류에 성공한 비율인 Sensitivity와 당뇨병이 있을 것이라 분류한 사람들 중 실제로 당뇨병을 가진 사람들의 비율인 Precision이 중요한 지표라 판단하였다.

따라서 일반적으로 평가지표로 선택되는 misclassification rate이 아닌 Sensitivity와 Precision의 조화평균인 f1-score를 최종적으로 Decision Tree 성능평가 기준에 가장 유의미한 지표라 판단하였다

Gini계수 기준 DT의 경우 Tree의 depth가 방대해지는 문제로 인해 Tree Depth 최대값을 1에서부터 10까지 설정하여 training data의 f1 score, test data의 f1 score를 구하였다.
Max depth에 따라 최적화된 모델을 구하기 위해 시각화를 진행한 결과는 다음과 같았다.



train data를 이용한 decision tree로 분류한 결과는 depth가 커질수록 f1 score의 값이 올라가는 것을 확인할 수 있었다. 하지만 test data의 경우 depth가 커질수록 f1 score의 값이 증가하는 경향을 보이지 않았다. 적절히 높은 f1 score를 가지면서 overfitting문제를 피하기 위해 train f1 score와 test f1 score간의 차이가 작은 depth = 3인 경우가 gini model에서 적합한 모델이라고 판단할 수 있었다. 하지만 max depth외에 다른 parameter들 또한 고려대상이기에 grid search를 이용한 hyperparameter tuning을 진행하였다.

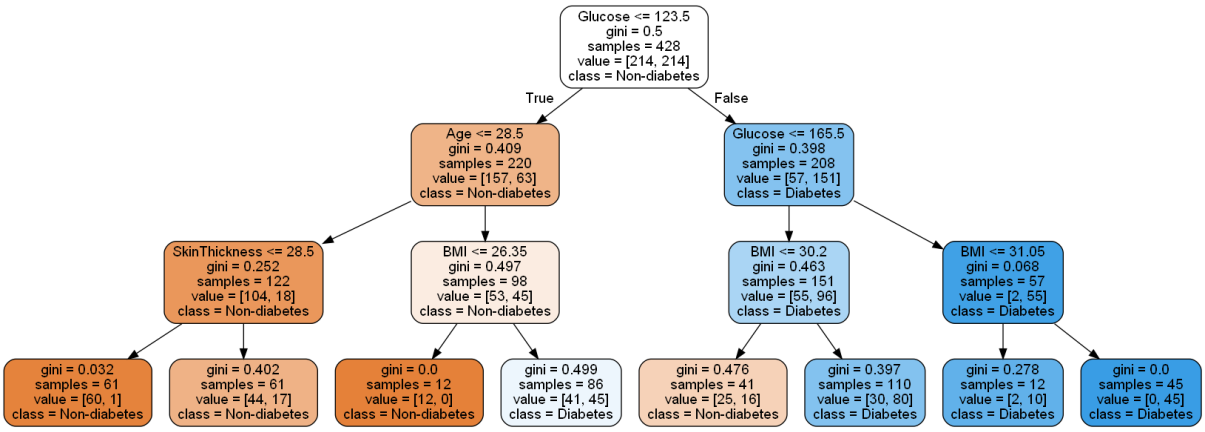
2.3 하이퍼 파라미터 튜닝

```
from sklearn.model_selection import GridSearchCV

param_range1 = [1,2,3,4,5,6,7,8,9,10] # max depth 1~10
param_range2 = [10, 20, 30, 40, 50] # 한 leaf에 sample을 최소 sample을 몇명 이상으로 할건지
param_range3 = ['gini'] # gini방법 혹은 entropy방법

param_grid = [{'decisiontreeclassifier__max_depth': param_range1,
               'decisiontreeclassifier__min_samples_leaf': param_range2,
               'decisiontreeclassifier__criterion': param_range3}] # 3가지 parameter 조합

gs = GridSearchCV(estimator = pipe_tree, # classification을 위한 tree
                  param_grid = param_grid, # 찾고자하는 파라미터. dictionary 형식
                  scoring = 'f1',
                  cv=10,
                  n_jobs= -1) # 병렬 처리갯수? -1은 전부를 의미

gs = gs.fit(train_features_us, train_target_us)
```

```
print(gs.best_score_) # gini를 사용한경우 가장 f1이 좋았을때, max depth가 3이었을때, min samples leaf가 10이었을때
print(gs.best_params_)
```

```
0.7391368722785278
{'decisiontreeclassifier__criterion': 'gini', 'decisiontreeclassifier__max_depth': 3, 'decisiontreeclassifier__min_samples_leaf': 10}
```

Gini계수를 기준으로 max_depth, min_samples_leaf라는 2가지 parameter를 고려해 최적의 f1 score를 가진 경우는 max_depth가 3, min_samples_leaf가 10인 경우로 나타났다. 이 모델을 시각화하면 다음과 같다.



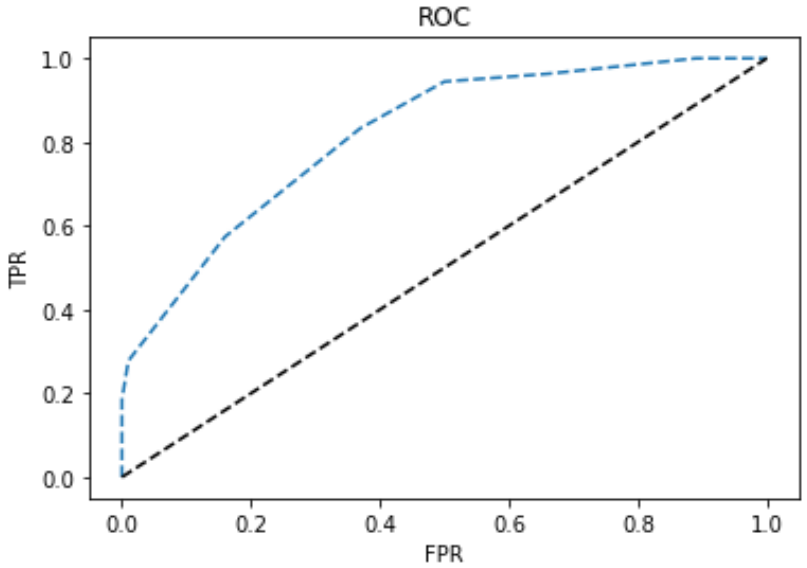
이 Decision Tree model로 test data를 분류하여 confusion matrix를 구하고 다음과 같은 measure score들과 ROC curve, 그에 따른 AUC값을 구하였다.

	Predict[0]	Predict[1]
True[0]	63	37
True[1]	9	45

Classification Report

	precision	recall	f1-score	support
0	0.88	0.63	0.73	100
1	0.55	0.83	0.66	54
accuracy			0.70	154
macro avg	0.71	0.73	0.70	154
weighted avg	0.76	0.70	0.71	154

```
Accuracy:  0.7013
Misclassification rate:  0.2987
Sensitivity(recall):  0.8333
Specificity:  0.6300
precision:  0.5488
f1_score:  0.6618
```

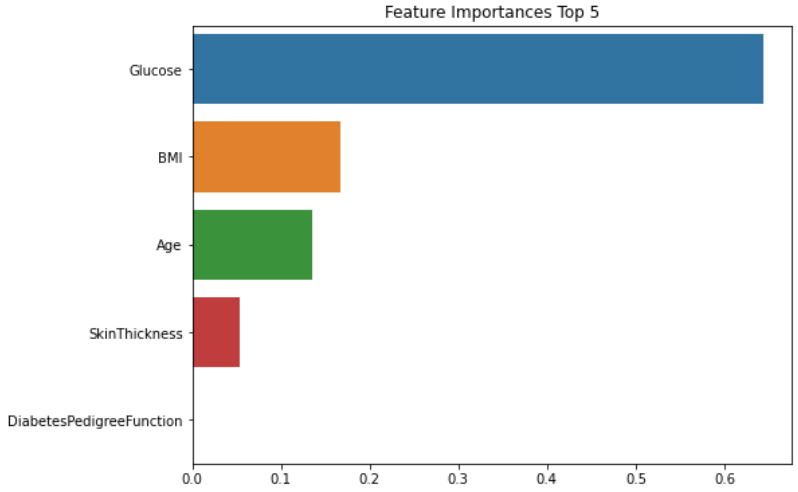


AUC: 0.732

앞에서 구한 Decision Tree 전체를 설명하기에는 무엇을 기준으로 삼아야 할지 모호하기

에 Tree를 만드는데 어떤 feature들이 얼마나 중요한지를 평가하는 feature importance를 사용할 수 있다. 이는 0과 1사이의 수이며, 특정 feature가 Tree model을 구성하는 비율을 나타낸다. 0에 가까울수록 사용되지 않는다는 의미이고, 1에 가까울수록 완벽하게 Target class를 예측했다는 의미를 가진다.

Gini 최적모델의 경우 Glucose(혈당), BMI(체질량지수), Age(나이), SkinThickness순으로 유의미하고 diabetes class를 예측하는데 중요한 역할을 한다고 판단할 수 있었다.



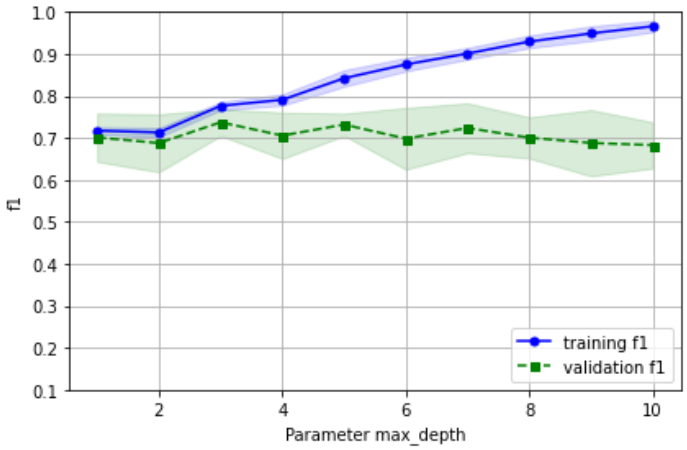
2.4 Entropy 기준 결정나무 적용

entropy를 기준으로 DT 모델링을 진행하기위해 DecisionTreeClassifier을 이용하였다. 그리고 10-fold의 Cross Validation을 진행하여 각 경우에서 test data의 f1 score값을 구하였다.

```
CV f1: [0.64       0.52941176 0.60465116 0.51428571 0.51162791 0.71111111
 0.55555556 0.72340426 0.72       0.7027027 ]
CV f1(Mean): 0.621 (std: 0.085)
```

Entropy기준 DT의 경우 역시 Tree의 depth가 방대해지는 문제로 인해 Tree Depth 최대값을 1에서부터 10까지 설정하여 training data의 f1 score, test data의 f1 score를 구하였다.

Max depth에 따라 최적화된 모델을 구하기 위해 시각화를 진행한 결과는 다음과 같았다.



train data를 이용한 decision tree로 분류한 결과는 depth가 커질수록 f1 score의 값이 올라가는 것을 확인할 수 있었다. 하지만 test data의 경우 depth가 커질수록 f1 score의 값이 증가하는 경향을 보이지 않았기에 적절히 높은 f1 score를 가지면서 overfitting문제를 피하기위해 train f1 score와 test f1 score간의 차이가 작은 depth = 4인 경우를 entropy model에서 또한 적합한 모델이라고 판단했다.

하지만 max depth외에 다른 parameter들 또한 고려대상이기에 grid search를 이용한 hyperparameter tuning을 진행하였다.

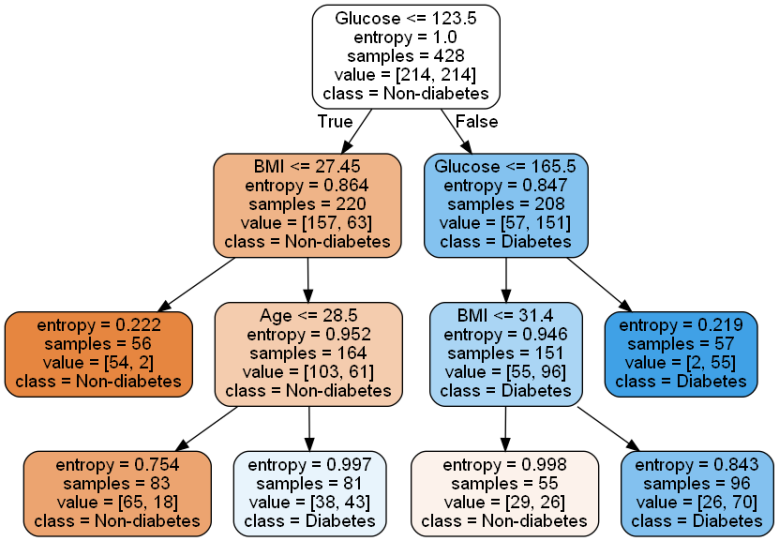
2.5 하이퍼 파라미터 튜닝

```
from sklearn.model_selection import GridSearchCV

param_range1 = [1,2,3,4,5,6,7,8,9,10] # max depth 1~10
param_range2 = [10, 20, 30, 40, 50] # 한 leaf에 sample을 최소 sample을 몇명 이상으로 할건지
param_range3 = ['entropy'] # gini방법 혹은 entropy방법

param_grid = [{'decisiontreeclassifier__max_depth': param_range1,
               'decisiontreeclassifier__min_samples_leaf': param_range2,
               'decisiontreeclassifier__criterion': param_range3}] # 3가지 parameter 조합

gs = GridSearchCV(estimator = pipe_tree, # classification을 위한 tree
                  param_grid = param_grid, # 찾고자하는 파라미터. dictionary 형식
                  scoring = 'f1',
                  cv=10,
                  n_jobs= -1) # 병렬 처리갯수? -1은 전부를 의미

gs = gs.fit(train_features_us, train_target_us)
```

```
print(gs.best_score_) # entropy를 사용한경우 가장 f1이 좋았을때, max depth가 4이었을때, min samples leaf가 50이었을때
print(gs.best_params_)
```

```
0.7473119207413134
{'decisiontreeclassifier__criterion': 'entropy', 'decisiontreeclassifier__max_depth': 4, 'decisiontreeclassifier__min_samples_leaf': 50}
```

entropy값을 기준으로 max_depth, min_samples_leaf라는 2가지 parameter를 고려해 최적의 f1 score를 가진 경우는 max_depth가 4, min_samples_leaf가 50인 경우로 나타났다. 이 모델을 시각화한 결과는 다음과 같다.



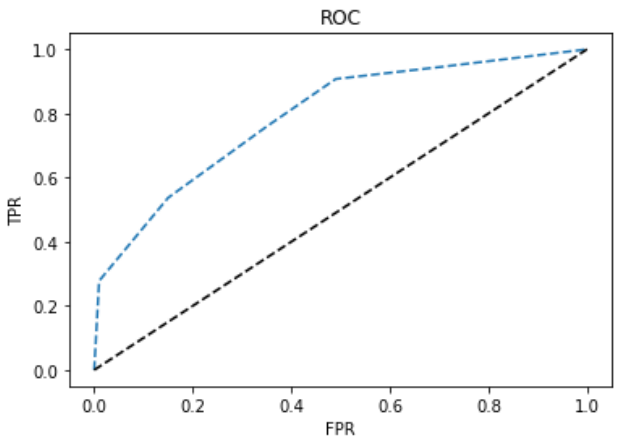
이 Decision Tree model로 test data를 분류하여 confusion matrix를 구하고 다음과 같은 measure score들과 ROC curve, 그에 따른 AUC값을 구했다.

	Predict[0]	Predict[1]
True[0]	65	35
True[1]	13	41

Classification Report

	precision	recall	f1-score	support
0	0.83	0.65	0.73	100
1	0.54	0.76	0.63	54
accuracy			0.69	154
macro avg	0.69	0.70	0.68	154
weighted avg	0.73	0.69	0.70	154

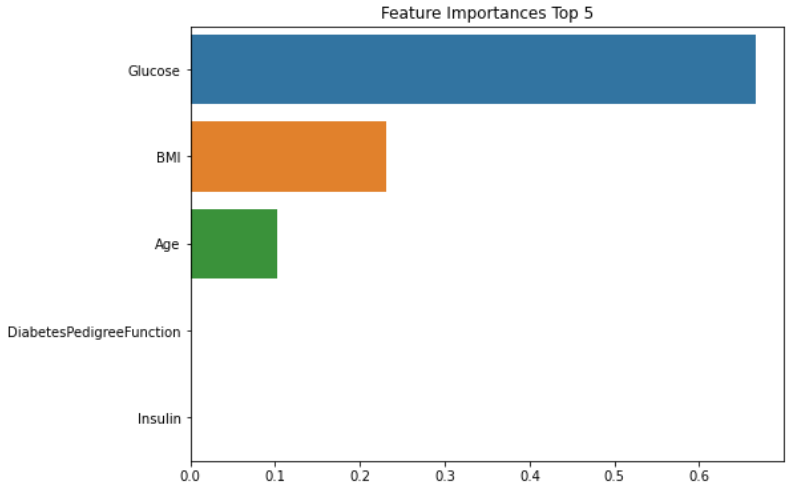
```
Accuracy:  0.6883
Misclassification rate:  0.3117
Sensitivity(recall):  0.7593
Specificity:  0.6500
precision:  0.5395
f1_score:  0.6308
```



AUC: 0.705

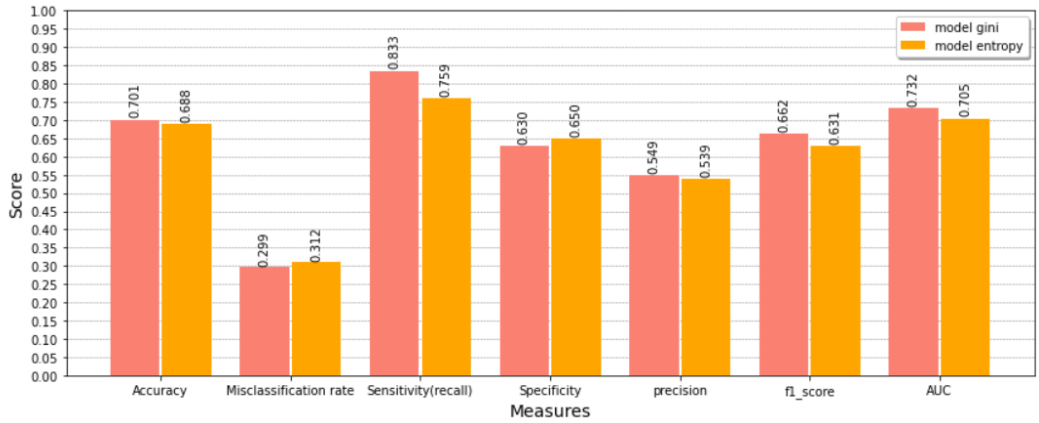
앞에서 구한 Decision Tree 전체를 설명하기에는 무엇을 기준으로 삼아야 할지 모호하기에 Tree를 만드는데 어떤 feature들이 얼마나 중요한지를 평가하는 feature importance를 사용하였다. 이를 통해 파악한 결과, Entropy 최적 모델의 경우 Glucose(혈당), BMI(체질

량지수), Age(나이)순으로 유의미하고 diabetes class를 예측하는데 중요한 역할을 한다고 판단할 수 있었다.



2.6 Gini index와 Entropy 기준 결과 비교

	Gini Index 기준	Entropy 기준
Accuracy	**0.701**	0.688
Misclassification	**0.299**	0.312
Recall	**0.833**	0.759
Specificity	0.630	**0.650**
Precision	**0.549**	0.539
F1-score	**0.7352**	0.705

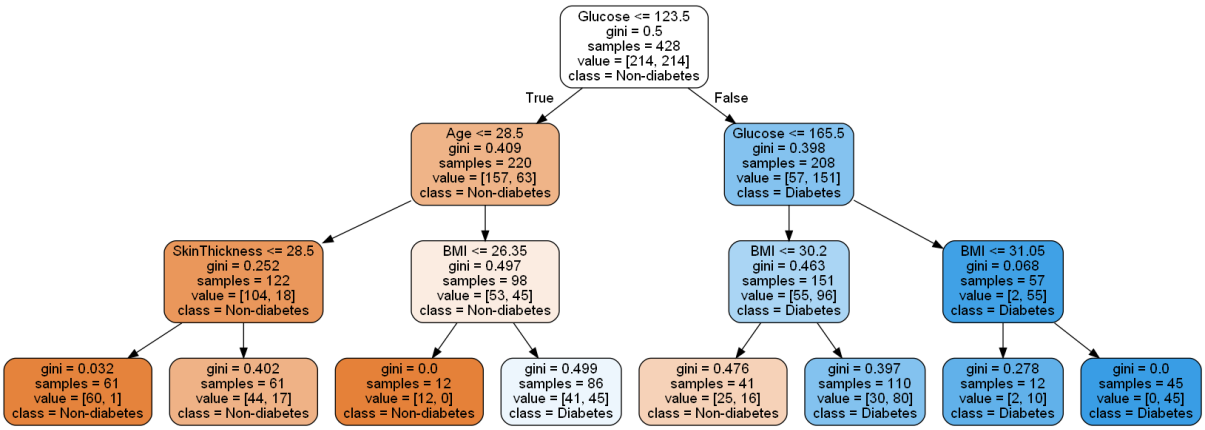


위 표와 그래프를 통해 Gini index를 기준으로 결정나무를 그렸을 때와 Entropy를 기준

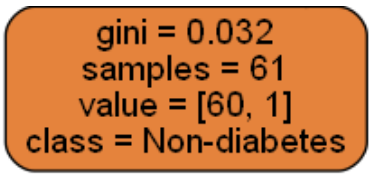
으로 결정나무를 그렸을 때의 성능을 비교하였다. Specificity를 제외한 대부분의 영역에서 Gini index를 기준으로 나누었을 때 더 성능이 뛰어난 것으로 나타났다. 특히 Classification DT에서 주로 성능 평가 지표로 선정되는 Misclassification rate과 본 데이터에 적합한 지표로 선정한 F1-score에서 Gini를 기준으로 했을 때 더 뛰어난 성능을 보이는 것으로 나타났다.

따라서 Gini Index를 기준으로 그린 의사결정나무를 바탕으로 규칙해석을 진행하였다.

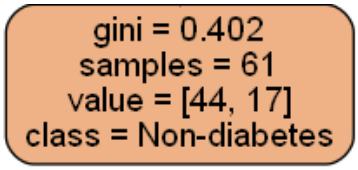
2.7 최종 해석



criterion = 'entropy', max_depth = 3, min_samples_leaf = 10인 모델을 기준으로 decision tree는 다음과 같이 나타난다. 도출된 decision tree에는 leaf node별 번호가 적혀있지 않아 편의상 왼쪽 노드부터 순서대로 1번으로 하여 해석을 진행하고자 한다.

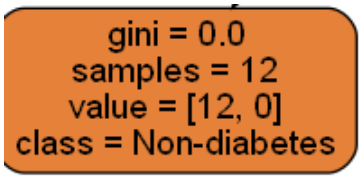


1번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이하 → 나이가 28.5이하 → 피부 두께가 28.5 이하의 규칙이 적용된 노드이다. 해당 노드는 총 61개의 sample들로 이루어져 있고, 60개의 non-diabetes sample과 1개의 diabetes sample로 이루어져 있기에 Non-diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 123.5이하, 나이가 28.5세 이하, 피부 두께가 28.5이하인 사람들은 Non-diabetes로 분류됨을 의미한다.

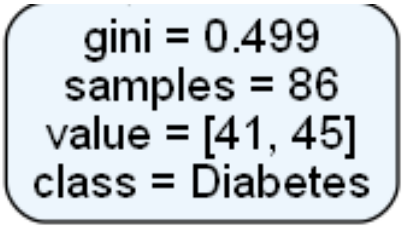


2번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이하 → 나이가 28.5세 이하 → 피부 두께

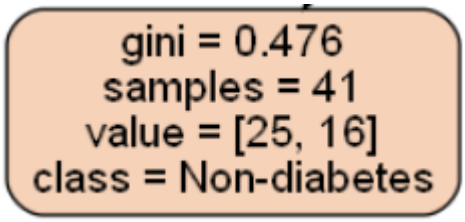
가 28.5 이상인 규칙이 적용된 노드이다. 해당 노드는 총 61개의 sample들로 이루어져 있고, 44개의 non-diabetes sample과 17개의 diabetes sample로 이루어져 있기에 Non-diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 123.5이하, 나이가 28.5세 이하, 피부 두께가 28.5이상인 사람들은 Non-diabetes로 분류됨을 의미한다.



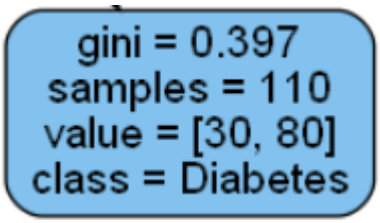
3번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이하 → 나이가 28.5세 이상 → BMI 지수가 26.35 이하인 규칙이 적용된 노드이다. 해당 노드는 총 12개의 sample들로 이루어져 있고, 12개의 non-diabetes sample과 0개의 diabetes sample로 이루어져 있기에 Non-diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 123.5이하, 나이가 28.5세 이상, BMI 지수가 26.35이하인 사람들은 Non-diabetes로 분류됨을 의미한다.



4번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이상 165.5이하 → BMI(체질량지수)가 31.4 이하의 규칙이 적용된 노드이다. 해당 노드는 총 57개의 sample들로 이루어져 있고, 30개의 non-diabetes sample과 27개의 diabetes sample로 이루어져 있기에 Non-diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 123.5이상 165.5이하 → BMI(체질량지수)가 31.4이하인 사람들은 Non-diabetes로 분류됨을 의미하지만 전체 대비 47%인 27명의 Diabetes가 포함되어 있기에 유의미한 분류는 아닌 것으로 판단된다.

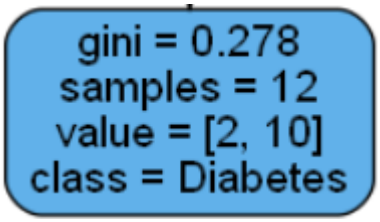


5번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이상 165.5이하 → BMI(체질량지수)가 30.2 이하의 규칙이 적용된 노드이다. 해당 노드는 총 41개의 sample들로 이루어져 있고, 25개의 non-diabetes sample과 16개의 diabetes sample로 이루어져 있기에 Non-diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 123.5이하 165.5이하, BMI(체질량지수)가 30.2이하인 사람들은 Non-diabetes로 분류됨을 의미한다.

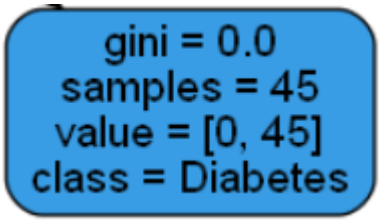


6번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이상 165.5이하 → BMI(체질량지수)가 30.2

이상의 규칙이 적용된 노드이다. 해당 노드는 총 110개의 sample들로 이루어져 있고, 30개의 non-diabetes sample과 80개의 diabetes sample로 이루어져 있기에 Diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 123.5이상 165.5이하, BMI(체질량지수)가 30.2이상인 사람들은 Diabetes로 분류됨을 의미한다.



7번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이상 → Glucose(혈당)이 165.5이상 → BMI(체질량지수)가 31.05이하의 규칙이 적용된 노드이다. 해당 노드는 총 12개의 sample들로 이루어져 있고, 2개의 non-diabetes sample과 10개의 diabetes sample로 이루어져 있기에 Diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 165.5이상, BMI(체질량지수)가 31.05이하인 사람들은 Diabetes로 분류됨을 의미한다.



8번 Leaf 노드는 상위에서부터 Glucose(혈당)이 123.5이상 → Glucose(혈당)이 165.5이상 → BMI(체질량지수)가 31.05이상의 규칙이 적용된 노드이다. 해당 노드의 45개의 sample들중 모두인 45개가 diabetes sample로 이루어져 있기에 Diabetes class로 분류된다. 결과적으로 Glucose(혈당)이 165.5이상, BMI(체질량지수)가 31.05이상인 사람들은 Diabetes로 분류됨을 의미한다.

3. Regression Tree

3.1. 데이터 설명 및 전처리

연속형변수의 의사결정나무를 위해서 Kaggle의 미국 king county 지역의 집값 데이터를 사용하였다. 해당 데이터는 2014년 5월부터 2015년 5월까지 Seattle을 포함한 King County 지역의 집값을 데이터로 갖으며, 총 21개의 변수 하에 21,613개의 집 값 데이터를 갖고 있다. 각각의 변수들이 집 값(price)에 어떠한 영향을 주는지에 대한 데이터이다. 총 20개의 독립변수와 1개의 종속변수로 구성되어있다. 독립변수 중 날짜, ID, 우편번호, 연도의 고유 값과 유무의 binary 변수를 제외하고는 모두 연속형 변수라고 할 수 있다. 아래 표는 변수에 대한 설명이다.

	Data Type	변수 설명

id	Int	해당 집의 고유 번호
date	Object	집 판매가 이루어진 날짜
Bedrooms	Int	침실의 개수
Bathrooms	Float	욕실의 개수
Sqft_living	int	집 내부의 면적
Sqft_lot	Int	집 외부의 면적
Floors	Float	층 수
Waterfront	Int	수도 Waterfornt 유무
View	Int	집 밖 풍경이 얼마나 좋은지에 대한 평가
Condition	Int	집의 상태가 얼마나 좋은지에 대한 평가
Grade	Int	건축과 디자인이 얼마나 좋은지에 대한 평가
Sqft_above	Int	집 위의 면적
Sqft_basement	Int	지하 공간의 면적
Yr_built	Int	초기 지어진 연도
Yr_renovated	Int	재건축된 연도
Zipcode	Int	해당 집이 위치한 지역의 우편번호
Lat	Float	위도
Long	Float	경도
Sqft_living15	Int	근처 15개의 집 내부 면적
Sqft_lot15	Int	근처 15개의 집 외부 면적
Price (Target)	**float**	**집이 판매된 가격**

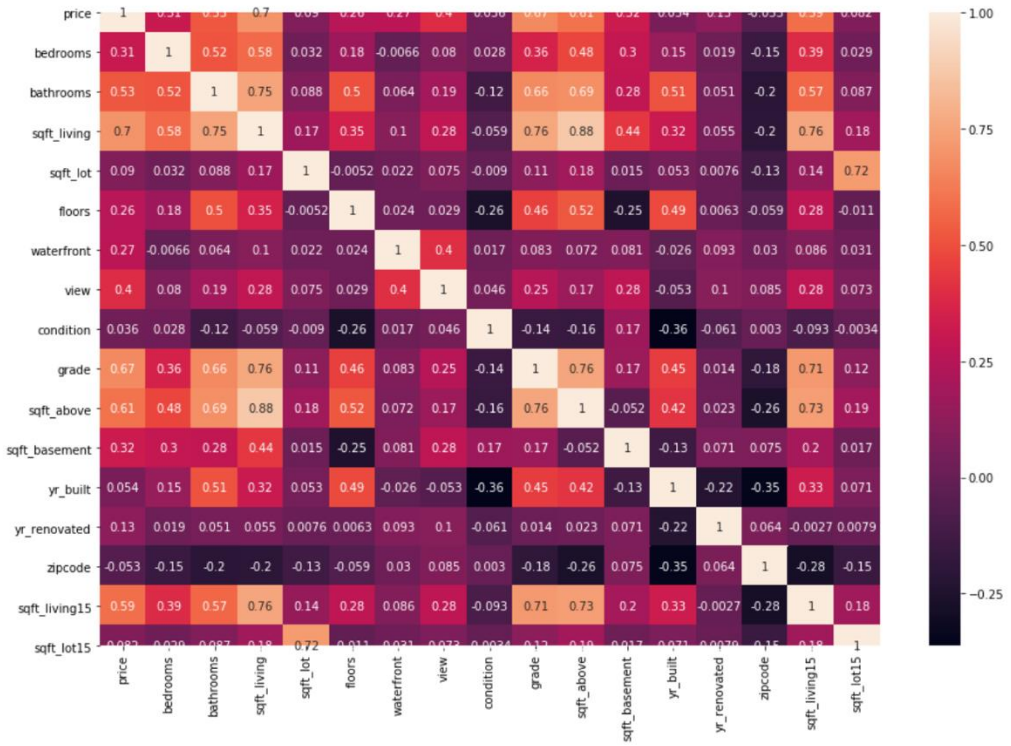
본격적인 의사결정나무를 적용하기 전, 먼저 적용할 변수를 골라낼 작업을 전처리로 진행하였다. 20개의 독립변수는 너무 많다고 판단하여, 변수를 줄이고자 하였다. 먼저 집의 고유 번호를 나타내는 id와 집판매가 이루어진 날짜인 date는 해당 과제를 진행하는데에 영향이 크지 않을 것으로 판단하여 제외하였다. 또한, Lat과 Long은 각각 위도와 경도를 나타내는데, King county라는 특정 지역의 집을 나타내며 또한 Zipcode라는 해당 지역을

나타내는 또 다른 변수가 있기 때문에 불필요하다고 판단하여 제외하였다.

price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement
221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0
538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400
180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0
604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910
510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0

yr_built	yr_renovated	zipcode	sqft_living15	sqft_lot15
1955	0	98178	1340	5650
1951	1991	98125	1690	7639
1933	0	98028	2720	8062
1965	0	98136	1360	5000
1987	0	98074	1800	7503

위의 표는 id, date, lat, long 네가지 데이터를 제외한 데이터이다. 제외한 이후 각 변수간의 상관관계를 시각화하여보기 위해 히트맵을 이용하여 정량적으로 표시해보았다. 양의 상관도가 높으면 색깔이 옅어지고, 음의 상관도가 높으면 색깔이 짙어지고, 아무런 상관관계가 없을수록 빨강색에 가까운 색으로 표현하였다.



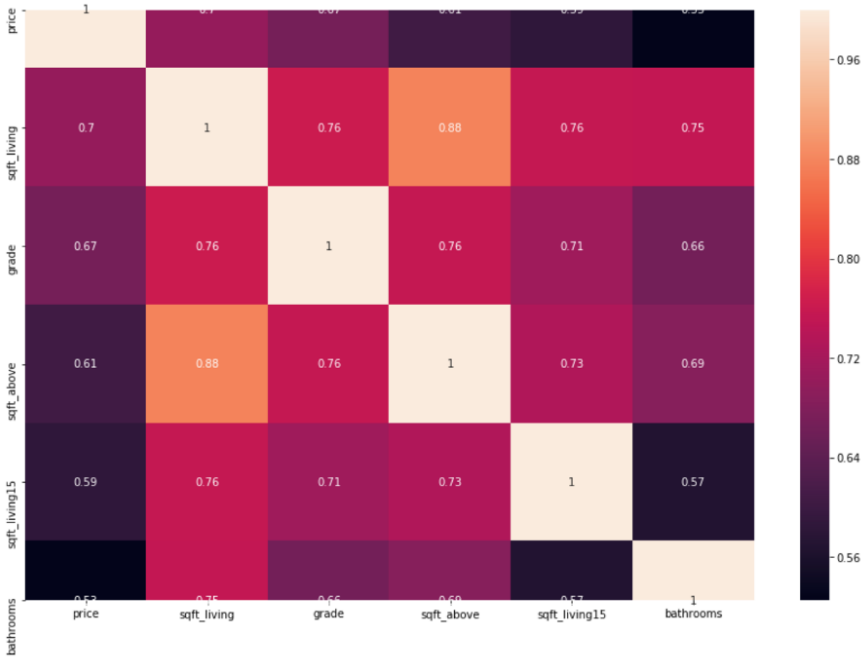
상관관계를 확인한 이유는 20개 중 4개를 제외한 16개의 변수로 의사결정나무를 적용하는 것은 변수가 너무 많다고 판단되었기 때문이다. 따라서 종속변수와의 상관도가 0.5이상인 독립변수만을 골라내어 적용하기로 하였다. 시각화를 통해서 확인한 결과, 집 내부의 면적을 의미하는 sqft_living, 집의 점수를 의미하는 grade, 옥상면적을 나타내는

sqft_above, 근처 15개 집의 내부 평균 면적을 의미하는 sqft_living15, 욕실 수를 의미하는 bathrooms 총 5개의 변수가 골라졌다. 이밖에도 yr_renovated 변수처럼 특별하게 확인이 가능한 경우도 있었다.

```
correlation = dataset.corr(method='pearson')
columns = correlation.nlargest(6, 'price').index
columns
```

```
Index(['price', 'sqft_living', 'grade', 'sqft_above', 'sqft_living15',
       'bathrooms'],
      dtype='object')
```

```
data = dataset.copy().loc[:,['price', 'sqft_living', 'grade', 'sqft_above'
, 'sqft_living15',
       'bathrooms']]
```



최종적으로 의사결정나무를 적용할 종속변수 1개와 독립변수 5개의 변수간의 상관도를 시각화한 히트맵이다. 앞선 히트맵과 같은 상관도를 확인할 수 있었다. 이 중 상관도가 가장 높은 것은 0.7로 sqft_living이었고, bathrooms이 0.53으로 제일 낮았다.

3.2 회귀나무 적용

의사결정나무 적용을 위해 전처리를 통해 결정된 6개의 변수가 있는 데이터를 독립변수와 종속변수 데이터로 분리하였다. 이후, Train set와 Test set을 8:2로 분리하여 확인하고자 하였다.

```
X = data.iloc[:,1:]
y = data.iloc[:,0]
```

```
X.head()
```

	sqft_living	grade	sqft_above	sqft_living15	bathrooms
0	1180	7	1180	1340	1.00
1	2570	7	2170	1690	2.25
2	770	6	770	2720	1.00
3	1960	7	1050	1360	3.00
4	1680	8	1680	1800	2.00

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

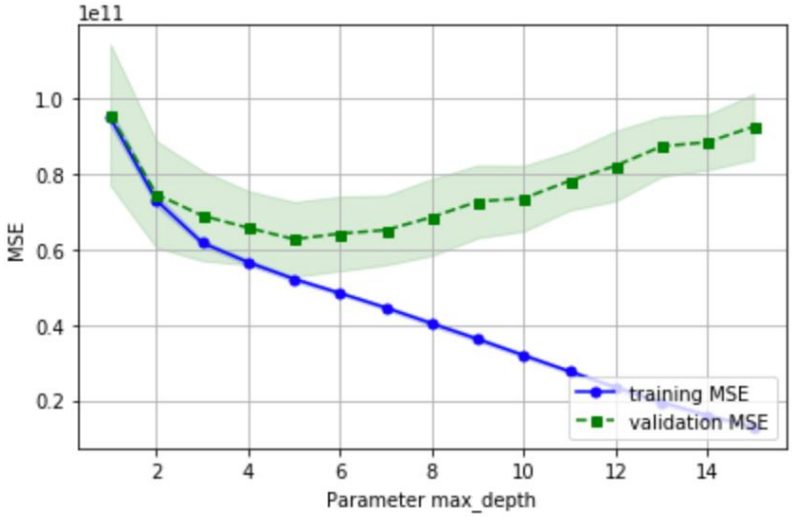
```
print(len(X_train))
print(len(y_train))
print(len(X_test))
print(len(y_test))
```

```
17290
17290
4323
4323
```

Train set와 Test set을 구분한 결과, 각각 17,290개, 4,323개의 데이터 개수임을 확인할 수 있었다.

1) Criterion = MSE인 경우

데이터를 구분한 이후에, Python의 DecisionTreeRegressor를 통해 의사결정나무 모델을 적용하였다. Tree의 Depth 최대값을 1부터 15까지 다르게 적용하였고, max depth에 따른 Train set과 Validation set의 MSE을 통해 성능비교를 하고자 하였다. 본 데이터에 적용하기 좋은 최적의 Depth를 알기 위해서는 그 Depth에 따른 MSE가 가장 작은 것이 좋겠지만, 과적합 문제 등을 위해 시각화로 그래프를 그려 확인하였다.

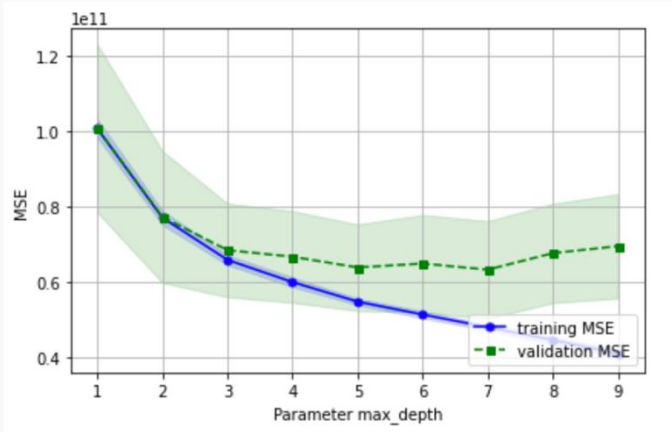


Train set의 mse는 Depth가 높아질수록 낮아진것을 확인할 수 있었지만, Validation set의

경우에는 5일때 MSE가 최저라는 것을 확인할 수 있었다. MSE가 낮으면 낮을수록 좋겠지만, 과적합 등의 문제가 있어 둘다 어느정도 낮은 MSE를 만족하는 Max_depth이 5일 때가 가장 적합한 모델임을 그래프를 통해서 확인할 수 있었다.

2) Criterion = MAE인 경우

다음은 분류기준을 MAE로 지정한 이후에 Tree의 Depth 최대값을 1부터 15까지 다르게 적용하였고, max depth에 따른 Train set과 Validation set의 MSE을 통해 성능비교를 하고자 하였다. 본 데이터에 적용하기 좋은 최적의 Depth를 알기 위해서는 그 Depth에 따른 MSE가 가장 작은 것이 좋겠지만, 과적합 문제 등을 위해 시각화로 그래프를 그려 확인하였다.



Train set의 mse는 Depth가 높아질수록 낮아진 것을 확인할 수 있었지만, Validation set의 경우에는 5와 7일때 MSE가 최저라는 것을 확인할 수 있었다. 두 수치 중 그래프를 통해 적절한 차이를 갖는 것은 max_depth이 5일때임을 알 수 있었다. 따라서 Max_depth이 5일 때가 가장 적합한 모델임을 그래프를 통해서 확인할 수 있었다.

3.4. 하이퍼 파라미터 튜닝

하이퍼 파라미터를 튜닝하기 위해 GridSearch를 이용했다.

DecisionTreeRegressor의 params는 다음과 같았다.

```
tr_regressor.get_params().keys()
```

```
dict_keys(['ccp_alpha', 'criterion', 'max_depth', 'max_features', 'max_leaf_nodes', 'min_impurity_decrease',
'min_impurity_split', 'min_samples_leaf', 'min_samples_split', 'min_weight_fraction_leaf', 'random_state', 's
plitter'])
```

다음은 각각의 params에 대해 설명을 한 것이다.

ccp_alpha : 최소 비용-복잡성 가지치기에 대한 하이퍼 파라미터

criterion : MSE 혹은 MAE 지수를 결정해주는 것으로 회귀 품질을 측정해준다.

max_depth : 트리의 최대 깊이

max_features : 각 노드에서 분할에 사용할 특징의 최대 수

min_impurity_decrease : 최소 불순도

splitter : 각 노드에서 분할을 선택하는 데 사용되는 전략

min_samples_split : 자식 노드를 분할하는 데 필요한 최소 샘플 수

min_samples_leaf : 리프 노드에 있어야 할 최소 샘플 수

min_weight_fraction_leaf : min_sample_leaf 와 같지만 가중치가 부여된 샘플 수에서의 비율

random_state : 난수 seed 설정

max_leaf_nodes : 리프 노드의 최대수

min_impurity_decrease : 최소 불순도

위의 max_depth 도출을 위한 검정곡선을 그릴 때와 마찬가지로 크게 분류기준을 MSE와 MAE로 구분하여 gridsearch를 진행하였다. 이때 분류기준 이외에 max_depth과 min_samples_leaf 파라미터를 각각 [1,2,3,4,5,6,7,8,9,10], min_samples_leaf는 [100, 200, 300, 400, 500]로 다르게 하여 gridsearch를 진행하였다.

1) Criterion = MSE인 경우

```
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
param_range1 = [1,2,3,4,5,6,7,8,9,10]
param_range2 = [100,200, 300, 400, 500]
param_range3 = ['mse'] # 'explained_variance'도 가능

param_grid = {'max_depth': param_range1,
              'min_samples_leaf': param_range2,
              'criterion': param_range3}

grid_cv = GridSearchCV(tr_regressor, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5, verbose=2)
grid_cv.fit(X_train, y_train)
print('GridSearchCV 최고 평균 정확도 수치: {:.4f}'.format(grid_cv.best_score_))
print('GridSearchCV 최적 하이퍼파라미터: ', grid_cv.best_params_)
```

```
print('GridSearchCV 최고 평균 정확도 수치: {:.4f}'.format(-grid_cv.best_score_))
print('GridSearchCV 최적 하이퍼파라미터: ', grid_cv.best_params_)
```

```
GridSearchCV 최고 평균 정확도 수치: 60634301262.7028
GridSearchCV 최적 하이퍼파라미터:  {'criterion': 'mse', 'max_depth': 10, 'min_samples_leaf': 100}
```

다음과 같은 결과가 도출되었다. 분류기준을 MSE 로 설정한 결과, 최적의 max_depth 은 10 이 나왔으며 min_samples_leaf 는 100 이 나왔다.

2) Criterion = MAE인 경우

```python
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
param_range1 = [1,2,3,4,5,6,7,8,9,10]
param_range2 = [100,200, 300, 400, 500]
param_range3 = ['mae'] # 'explained_variance'도 가능

param_grid = {'max_depth': param_range1,
              'min_samples_leaf': param_range2,
              'criterion': param_range3}

grid_cv = GridSearchCV(tr_regressor, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5, verbose=2)
grid_cv.fit(X_train, y_train)
print('GridSearchCV 최고 평균 정확도 수치: {:.4f}'.format(grid_cv.best_score_))
print('GridSearchCV 최적 하이퍼파라미터: ', grid_cv.best_params_)
```

```
GridSearchCV 최고 평균 정확도 수치: -62958137150.0322
GridSearchCV 최적 하이퍼파라미터:  {'criterion': 'mae', 'max_depth': 8, 'min_samples_leaf': 100}
```

다음과 같은 결과가 도출되었다. 분류기준을 MAE 로 설정한 결과, 최적의 max_depth 은 8 이 나왔으며 min_samples_leaf 는 100 이 나왔다.

➔ Grid Search 를 통하여 하이퍼파라미터 를 critertion 은 MSE 와 MAE 로 크게 분류하고, 각각에 맞게 max_depth 와 min_samples_leaf 를 다르게 하여 최적의 하이퍼파라미터를 도출하였다. 이때, 위의 각각 검정곡선을 그렸을 때의 max_depth 과는 다르게 나와 최종적으로 각각 모델의 R-squared 와 MSE 를 통해 최종 모델을 결정하고자 하였다.

3.4.1. 성능 평가

Criterion	Max_depth	Min_samles_leaf	R-sqaured	MSE
MSE	**5**	**100**	**0.562**	**52071149399.867**
		200	0.543	54381820844.142
		300	0.530	55847306839.897
	10	100	0.575	50523789576.605
MAE	5	100	0.546	54034645088.237
		200	0.530	55860559613.266
		300	0.512	58001871899.183
	8	100	0.562	52143574006.157

먼저 MSE 일 때는 Max_depth 이 10 일 때가 R-sqauared 값도 높고, MSE 도 낮아 적절한 모델로 보인다. 하지만, 일전의 그래프를 확인했듯이 max_depth 이 10 인 경우에는 격차가 너무 커지는 것을 볼 수 있는데 이는 과적합되는 모델이라고 판단할 수 있다. 따라서 분류기준이 MSE 일 때는 Max_depth 이 5 일 때가 적절하다고 판단하였다. 또한, 최소 leaf 샘플 수가 올라갈 수록 R-squared 값이 낮아지기 때문에 100 일 때가 최적임을 결론내었다.

MA 또한 튜닝을 통해 나온 Max_depth 인 8 일 때가 가장 적절해 보이지만, 검정곡선을 통해 확인할 수 있듯이 이는 과적합이라고 판단할 수 있었다. 따라서 분류기준이 MAE 일 때 Max_depth 이 5 일 때가 적절하다고 판단하였고, 이에 따라 샘플 수가 100 일 때가 최적임을 결론을 내었다.

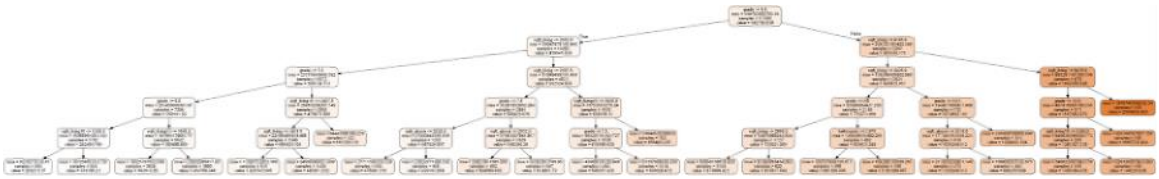
최종 두가지 중에서는 R-sqaured 이 더 높은 분류기준이 MSE 고 최대 depth 가 5 이고 min_sample-leaf 이 100 일때가 가장 최적임을 알 수 있었다. 따라서 본 데이터로 회귀나무를 적용했을 때는 다음의 모델이 가장 좋은 모델임으로 결론을 내었다.

3.5. 최종 해석

- 최종 모델: Criterion = MSE, Max_depth = 5, Min_samples_leaf = 100

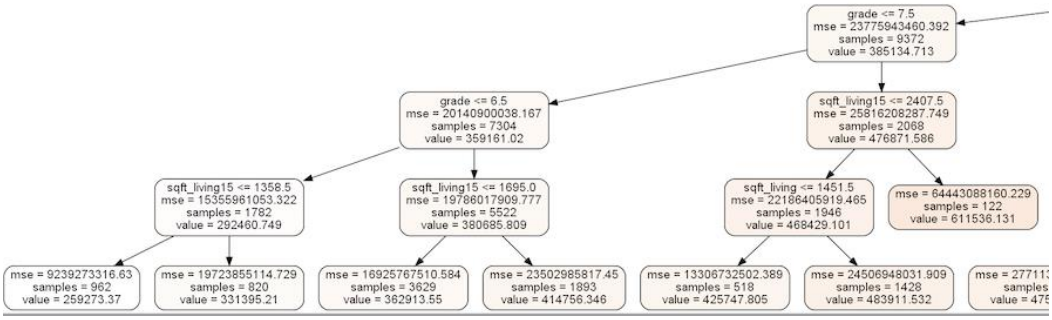
- **Node 규칙해석**

최종적으로 결정된 max_depth = 5를 기반으로 적용하였을 때의 나무 모양은 다음과 같다. Root Node는 Grade의 값이 8.5를 기준으로 하는 것으로 시작되었으며 17,290개의 sample로 시작되었다. 다음 node는 Root가 True일 경우 sqft_living이 2,032이하였고, False의 경우 Sqft_living이 4,185이하였다. 이하 노드와 규칙해석은 다음과 같다. 도출된 나무에는 leaf node별 번호가 적혀있지 않아 편의상 왼쪽부터 1번으로 하여 해석을 진행하고자 한다.



1) Grade <= 8.5 True인 경우

a. Sqft_living <= 2032.0 True인 경우



- *1번노드:*
  위 두 가지가 적용된 이후에, grade가 7.5이하 → grade가 6.5 이하 →

sqft_living15이 1358.5 이하의 규칙이 적용된 노드이다. 해당 노드는 회귀나무 모델이 적용된 Train set의 17290개의 데이터 중 962개의 샘플 수를 갖는다. Value값은 259273.37인 것으로 확인되었다. 결과적으로는 King County 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032 이하이고, 그 중 점수가 6.5이하 이고 주변 15개의 집 내부 면적 평균이 1358.5 이하인 집들의 판매된 가격 평균이 $259273.37임을 의미한다.

- *2번노드:*

  위 두 가지가 적용된 이후에, grade가 7.5이하 → grade가 6.5 이하 → sqft_living15이 1358.5 초과의 규칙이 적용된 노드이다. 해당 노드는 모델의 17290개의 데이터 중 820개의 샘플 수를 갖는다. Value값은 331395.21인 것으로 확인되었다. 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 점수가 6.5이하 이고 주변 15개의 집 내부 면적 평균이 1358.5 초과인 집들의 판매된 가격 평균이 $331395.21임을 의미한다.

- *3번노드:*

  위 두 가지가 적용된 이후에, grade가 7.5이하 → grade가 6.5 초과 → sqft_living15이 1695이하 규칙이 적용된 노드이다. 해당 노드는 모델의 17290개의 데이터 중 3629개의 샘플 수를 갖는다. Value값은 362913.55인 것으로 확인되었다. 결과적으로 해당 지역의 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 점수가 6.5초과이고 집 주변 15개의 집 내부 면적 평균이 1695 이하인 집들의 판매된 가격 평균이 $362913.55임을 의미한다.

- *4번노드:*

  위 두 가지가 적용된 이후에, grade가 7.5이하 → grade가 6.5 초과 → sqft_living15이 1695초과 규칙이 적용된 노드이다. 해당 노드는 모델의 17290개의 데이터 중 1893개의 샘플 수를 갖는다. Value값은 414756.346인 것으로 확인되었다. 결과적으로 해당 지역의 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 점수가 6.5초과이고 집 주변 15개의 집 내부 면적 평균이 1695 초과인 집들의 판매된 가격 평균이 $414756.346임을 의미한다.

- *5번노드:*

  위 두 가지가 적용된 이후에, grade가 7.5초과 → sqft_living15이 2407.5 이하 → sqft_living이 1451.5이하인 규칙이 적용된 노드이다. 해당 노드는 모델의 17290개의 데이터 중 518개의 샘플 수를 갖는다. Value값은 425757.805인 것으로 확인되었다. 결과적으로 해당 지역의 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, grade가 7.5초과이고, 주변 15개의 집 내부 평균 면적이 2407.5이하이고, 집 내부의 면적이 1451.5이하인 집들의 판매된 가격 평균이 $425757.805임을 의미한다.

- *6번노드:*

  위 두 가지가 적용된 이후에, grade가 7.5초과 → sqft_living15이 2407.5 이하 → sqft_living이 1451.5초과인 규칙이 적용된 노드이다. 해당 노드는 모델의 17290

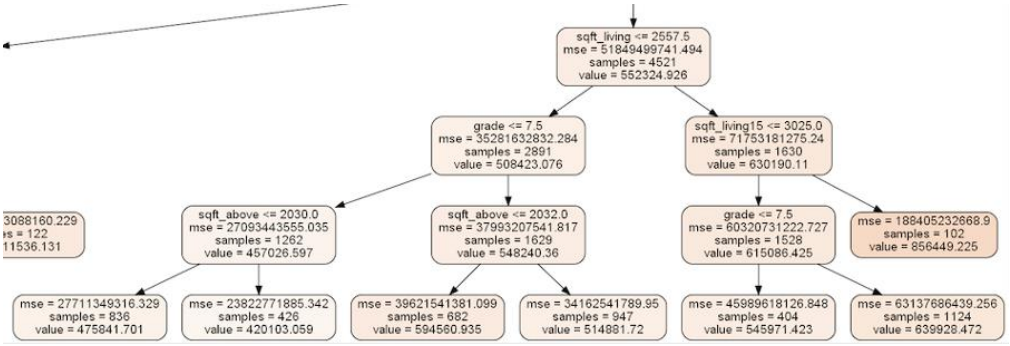
개의 데이터 중 1428개의 샘플 수를 갖는다. Value값은 483911.532인 것으로 확인되었다. 결과적으로 해당 지역의 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, grade가 7.5초과이고, 주변 15개의 집 내부 평균 면적이 2407.5이하이고, 집 내부의 면적이 1451.5초과이고 2032이하인 집들의 판매된 가격 평균이 $483911.532임을 의미한다.

- *7번노드:*

  위 두 가지가 적용된 이후에, grade가 7.5초과 → sqft_living15이 2407.5초과인 규칙이 적용된 노드이다. 해당 노드는 모델의 17290개의 데이터 중 122개의 샘플 수를 갖는다. Value값은 611536.131인 것으로 확인되었다. 결과적으로 해당 지역의 판매된 집 중 grade가 8.5이하인 집들 중 집 내부 면적이 2032 이하인 집들 중, grade가 7.5초과이고, 집 주변 15개의 집 내부 평균 면적이 2407.5초과인 집들의 판매된 가격 평균이 $611536.131임을 의미한다.

-

b. Sqft_living <= 2032.0 False인 경우



- *8번노드:*

  위 두 가지가 적용된 이후에, sqft_living이 2557.5이하 → grade가 7.5이하 → sqft_above가 2030이하인 규칙이 적용된 노드이다. 해당 노드는 회귀나무 모델이 적용된 Train set의 17290개의 데이터 중 836개의 샘플 수를 갖는다. Value값은 475841.701인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5이하인 집 중 grade가 7.5이하인 집 중 옥상 면적이 2030이하인 집들의 판매된 가격 평균이 $475841.701임을 의미한다.

- *9번노드:*

  위 두 가지가 적용된 이후에, sqft_living이 2557.5이하 → grade가 7.5이하 → sqft_above가 2030초과인 규칙이 적용된 노드이다. 해당 노드는 모델이 적용된 17290개의 데이터 중 426개의 샘플 수를 갖는다. Value값은 420103.059인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5이하인 집 중 grade가 7.5이하인 집 중 옥상 면적이 2030초과인 집들의 판매된 가격 평균이 $420103.059임을 의미한다.

- *10번노드:*

위 두 가지가 적용된 이후에, sqft_living이 2557.5이하 → grade가 7.5초과 → sqft_above가 2032이하인 규칙이 적용된 노드이다. 해당 노드는 모델이 적용된 17290개의 데이터 중 682개의 샘플 수를 갖는다. Value값은 594560.935인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5이하인 집 중 grade가 7.5초과인 집 중 옥상 면적이 2032이하인 집들의 판매된 가격 평균이 $594560.935임을 의미한다.

- *11번노드:*

  위 두 가지가 적용된 이후에, sqft_living이 2557.5이하 → grade가 7.5초과 → sqft_above가 2032초과인 규칙이 적용된 노드이다. 해당 노드는 947개의 샘플 수를 갖는다. Value값은 514881.72인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5이하인 집 중 grade가 7.5초과인 집 중 옥상 면적이 2032초과인 집들의 판매된 가격 평균이 $514881.72임을 의미한다.

- *12번노드:*

  위 두 가지가 적용된 이후에, sqft_living이 2557.5초과 → sqft_living15가 3025이하 → grade가 7.5이하인 규칙이 적용된 노드이다. 해당 노드는 404개의 샘플 수를 갖는다. Value값은 545971.423인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5초과인 집 중 집 주변 15개 집의 평균 내부 면적이 3025이하인 집 중 grade가 7.5이하인 집들의 판매된 가격 평균이 $545971.423임을 의미한다.

- *13번노드:*

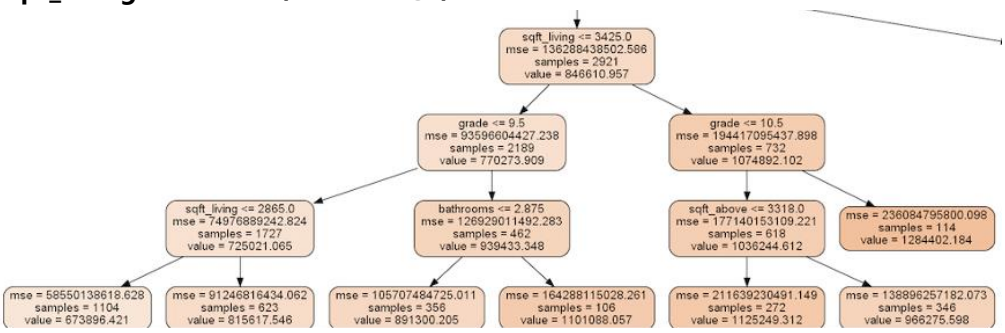
  위 두 가지가 적용된 이후에, sqft_living이 2557.5초과 → sqft_living15가 3025이하 → grade가 7.5초과인 규칙이 적용된 노드이다. 해당 노드는 1124개의 샘플 수를 갖는다. Value값은 639928.472인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5초과인 집 중 집 주변 15개 집의 평균 내부 면적이 3025이하인 집 중 grade가 7.5초과인 집들의 판매된 가격 평균이 $639928.472임을 의미한다.

- *14번노드:*

  위 두 가지가 적용된 이후에, sqft_living이 2557.5초과 → sqft_living15가 3025 인 규칙이 적용된 노드이다. 해당 노드는 102개의 샘플 수를 갖는다. Value값은 856449.225인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5이하인 집들 중 집 내부의 면적이 2032이하이고, 그 중 내부 면적이 2557.5초과인 집 중 집 주변 15개 집의 평균 내부 면적이 3025초과인 집들의 판매된 가격 평균이 $856449.225임을 의미한다.

2) **Grade <= 8.5 False인 경우**

a. **Sqft_living <= 4185가 True인 경우**



- *15번노드:*

  위 두 가지가 적용된 이후에 sqft_living이 3425이하 → grade가 9.5이하 → sqft_living이 2865이하인 규칙이 적용된 노드이다. 해당 노드는 1104개의 샘플 수를 갖는다. Value값은 673896.421인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425이하인 집 중 grade가 9.5이하인 집 중 집 내부 면적이 2865이하인 집들의 판매된 가격 평균이 $673896.421임을 의미한다.

- *16번노드:*

  위 두 가지가 적용된 이후에 sqft_living이 3425이하 → grade가 9.5이하 → sqft_living이 2865초과인 규칙이 적용된 노드이다. 해당 노드는 623개의 샘플 수를 갖는다. Value값은 815617.546인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425이하인 집 중 grade가 9.5이하인 집 중 집 내부 면적이 2865초과인 집들의 판매된 가격 평균이 $815617.546임을 의미한다.

- *17번노드:*

  위 두 가지가 적용된 이후에 sqft_living이 3425이하 → grade가 9.5초과 → bathrooms이 2.875이하인 규칙이 적용된 노드이다. 해당 노드는 356개의 샘플 수를 갖는다. Value값은 891300.205인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425이하인 집 중 grade가 9.5초과인 집 중 욕실 개수가 2.875이하인 집들의 판매된 가격 평균이 $891300.205임을 의미한다.

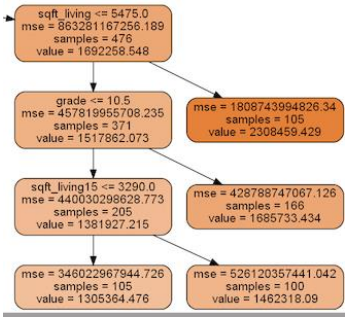
- *18번노드:*

  위 두 가지가 적용된 이후에 sqft_living이 3425이하 → grade가 9.5초과 → bathrooms이 2.875초과인 규칙이 적용된 노드이다. 해당 노드는 106개의 샘플 수를 갖는다. Value값은 1101088.057인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425이하인 집 중 grade가 9.5초과인 집 중 욕실 개수가 2.875초과인 집들의 판매된 가격 평균이 $1101088.057임을 의미

한다.

- *19번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 3425초과 → grade가 10.5이하→ sqft_above가 3318이하인 규칙이 적용된 노드이다. 해당 노드는 272개의 샘플 수를 갖는다. Value값은 1125249.312인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425초과인 집 중 grade가 10.5이하인 집 중 옥상 면적이 3318이하인 집들의 판매된 가격 평균이 $1125249.312임을 의미한다.
- *20번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 3425초과 → grade가 10.5이하→ sqft_above가 3318초과인 규칙이 적용된 노드이다. 해당 노드는 346개의 샘플 수를 갖는다. Value값은 966275.598인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425초과인 집 중 grade가 10.5이하인 집 중 옥상 면적이 3318초과인 집들의 판매된 가격 평균이 $966275.598임을 의미한다.
- *21번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 3425초과 → grade가 10.5 초과인 규칙이 적용된 노드이다. 해당 노드는 114개의 샘플 수를 갖는다. Value값은 1284402.184인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185이하이고, 그 중 내부 면적이 3425초과인 집 중 grade가 10.5 초과인 집들의 판매된 가격 평균이 $1284402.184임을 의미한다.

b. Sqft_living <= 4185가 False인 경우



- *22번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 5475이하 → grade가 10.5 이하 → sqft_living15가 3290이하인 규칙이 적용된 노드이다. 해당 노드는 105개의 샘플 수를 갖는다. Value값은 1305364.476인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185초과이고, 그 중 내부 면적이 5475이하인 집 중 grade가 10.5 이하인 집 중 주변 15개 집의 평균 내부 면적이 3290이하인 집들의 판매된 가격 평균이 $1305364.476임을 의미한다.

- *23번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 5475이하 → grade가 10.5 이하 → sqft_living15가 3290초과인 규칙이 적용된 노드이다. 해당 노드는 100개의 샘플 수를 갖는다. Value값은 1462318.09인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185초과이고, 그 중 내부 면적이 5475이하인 집 중 grade가 10.5 이하인 집 중 주변 15개 집의 평균 내부 면적이 3290초과인 집들의 판매된 가격 평균이 $1462318.09임을 의미한다.
- *24번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 5475이하 → grade가 10.5 초과인 규칙이 적용된 노드이다. 해당 노드는 166개의 샘플 수를 갖는다. Value값은 1685733.434인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185초과이고, 그 중 내부 면적이 5475이하인 집 중 grade가 10.5 초과인 집들의 판매된 가격 평균이 $1685733.434임을 의미한다.
- *25번노드:*
  위 두 가지가 적용된 이후에 sqft_living이 5475초과인 규칙이 적용된 노드이다. 해당 노드는 105개의 샘플 수를 갖는다. Value값은 2308459.429인 것으로 확인되었다. 결과적으로는 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185초과이고, 그 중 내부 면적이 5475 초과인 집들의 판매된 가격 평균이 $2308459.429임을 의미한다.

3) 최종 비교

총 25개의 Leaf node들의 샘플 수와 value 값을 정리한 표이다.

Node #	Sample 수	Value 값
1	962	$259,273.37
2	820	$331,395.21
3	3629	$362,913.55
4	1893	$414,756.35
5	518	$425,757.81
6	1428	$483,911.53
7	122	$611,536.13
8	836	$475,841.70
9	426	$420,103.06
10	682	$594,560.94
11	947	$514,881.72
12	404	$545,971.42
13	1124	$639,928.47
14	102	$856,449.23
15	1104	$673,896.42
16	623	$815,617.55
17	356	$891,300.21
18	106	$1,101,088.06
19	272	$1,125,249.31
20	346	$966,275.60
21	114	$1,284,402.18
22	105	$1,305,364.48
23	100	$1,462,318.09
24	166	$1,685,733.43
25	105	$2,308,459.43

먼저 가장 많은 sample 수를 갖는 Node는 3번 node로, 모델의 대상이 된 Train set의 17290개의 데이터 중 3629개의 데이터가 해당하는 node였다. 해당 노드의 규칙은 grade가 8.5이하 → 집 내부 면적이 2032 이하 → grade가 7.5이하 → grade가 6.5 초과 → sqft_living15이 1695이하이다. 결과적으로 King County 지역에 2014년 5월부터 2015년 5월까지 판매된 집들 중 8.5점 이하인 집 중, 집 내부 면적이 2032 이하이고 그 중 점수가 6.5초과 7.5이하인 집들 중 주변 15개 집의 내부 평균 면적이 1695이하인 집으로 골라내었을 때 가장 많은 수를 도출할 수 있다는 것이었다.

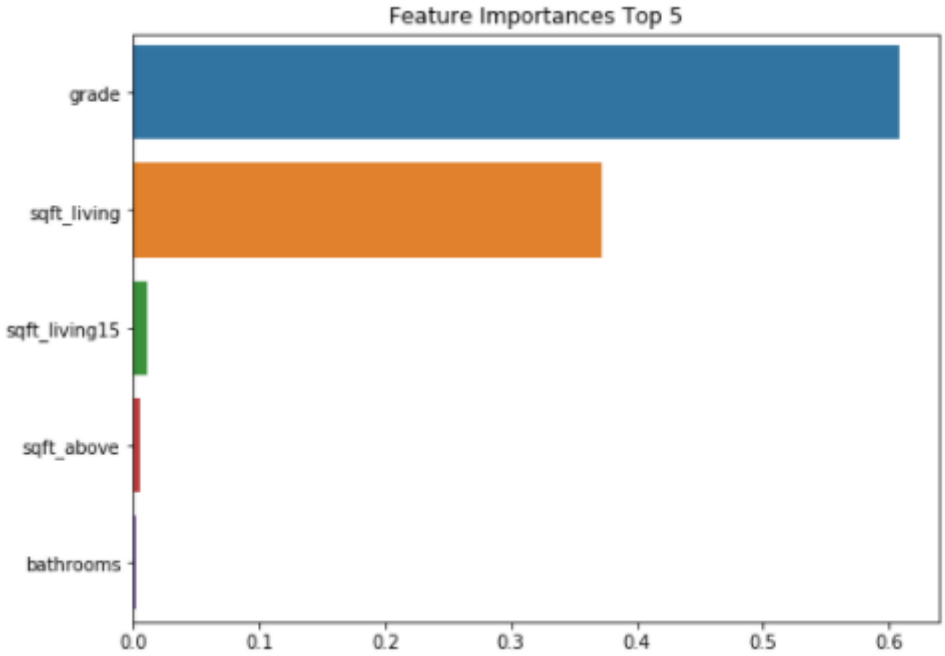
또한 Value값이 가장 큰 node는 마지막 25번 node 였으며, 규칙이 적용되는 105개의 샘플이 평균 $2308459.429 판매가격을 알 수 있다. 즉, 해당 지역의 정해진 기간에 판매된 집 중 grade가 8.5초과인 집들 중 집 내부의 면적이 4185초과이고, 그 중 내부 면적이 5475 초과인 집들의 판매된 가격 평균이 가장 높음을 알 수 있었다.

또한 표를 통해 한눈에 확인할 수 있듯이, Root node가 점수 8.5이하인 것이 기준이 되었는데, 8.5가 이하인 node들은 전반적으로 초과인 node들에 비해 value값이 낮은 것을 확인할 수 있었다.

3.6. Feature Importance 분석

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

```
grade          0.609096
sqft_living    0.371988
sqft_living15  0.010929
sqft_above     0.005338
bathrooms      0.002649
dtype: float64
```



DecisionTreeRegressor 를 한 결과, Grade, Sqft_living, Sqft_living15, Sqft_above, bathrooms 순으로 FeatrueImportance 가 나타났다. 시각화 내용을 살펴보면, Grade 와 Sqft_living 을 통해 Node 를 2 번 정도 나누는 데 사용하는 것을 볼 수 있었다. 트리가 그 특성을 많이 사용하고 클래스를 잘 나누는 특성이라는 것을 확인할 수 있었다. 다른 sqft_living15, sqft_above, bathrooms 의 특성을 Grade, Sqft_living 가 많이 담고 있다고도 볼 수 있다. 혹은 단지 그 특성을 트리가 택하지 않았을 수도 있다 해석이 된다.

4. Reference

categorical data : https://www.kaggle.com/uciml/pima-indians-diabetes-database

continuous data : https://www.kaggle.com/harlfoxem/housesalesprediction