의료인공지능

머신러닝 - 결정트리

고려대학교 의료빅데이터연구소
채민수(minsuchae@korea.ac.kr)

1. 결정 트리

◦ 결정 트리

- 전문가 시스템과 유사

- 데이터로부터 규칙을 학습함

- 불순도를 통해 특성과 임계치를 설정함

1. 결정 트리

- 결정 트리 학습 과정
  - 현재의 노드의 데이터에서 특성과 임계치를 사용하여 두 개의 서브셋으로 나눔
  - CART 알고리즘을 통해 비용 함수를 계산하고, CART 알고리즘의 비용이 가장 낮은 특성과 임계치를 구함
  - 자식 노드를 순회하며 반복
  - 최대 깊이 혹은 불순도의 값이 0이면 학습 종료

1. 결정 트리

◦ CART(Classification and regression tree)

$$J(k, k_t) = \frac{m_{left}}{m_{total}} I_{left} - \frac{m_{right}}{m_{total}} I_{right}$$

- 지니 불순도

$$I_G(i) = \sum_{k=1}^{classes} p(i,k)\big(1 - p(i,k)\big)$$

$$= \sum_{i=1}^{classes} p(i,k) - \sum_{i=1}^{classes} p(i,k)^2$$

$$= 1 - \sum_{i=1}^{classes} p(i,k)^2$$

1. 결정 트리

- CART(Classification and regression tree)

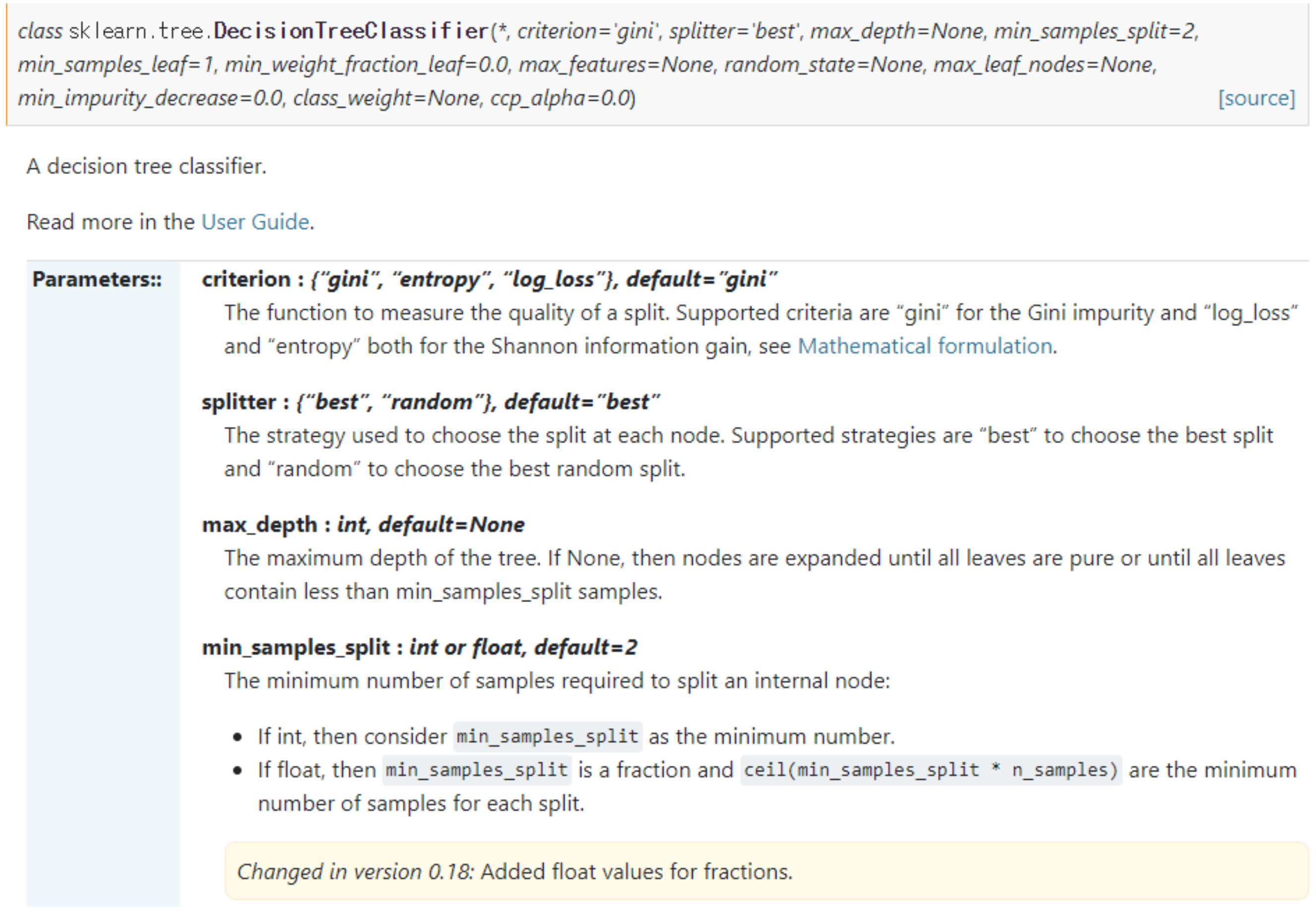
$$J(k, k_t) = \frac{m_{left}}{m_{total}} I_{left} - \frac{m_{right}}{m_{total}} I_{right}$$

- 엔트로피

$$I_H(i) = \sum_{k=1}^{classes} p(i,k) \log_2 p(i,k)$$

1. 결정 트리

◦ 결정 트리 분류



class sklearn.tree.**DecisionTreeClassifier**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters::

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

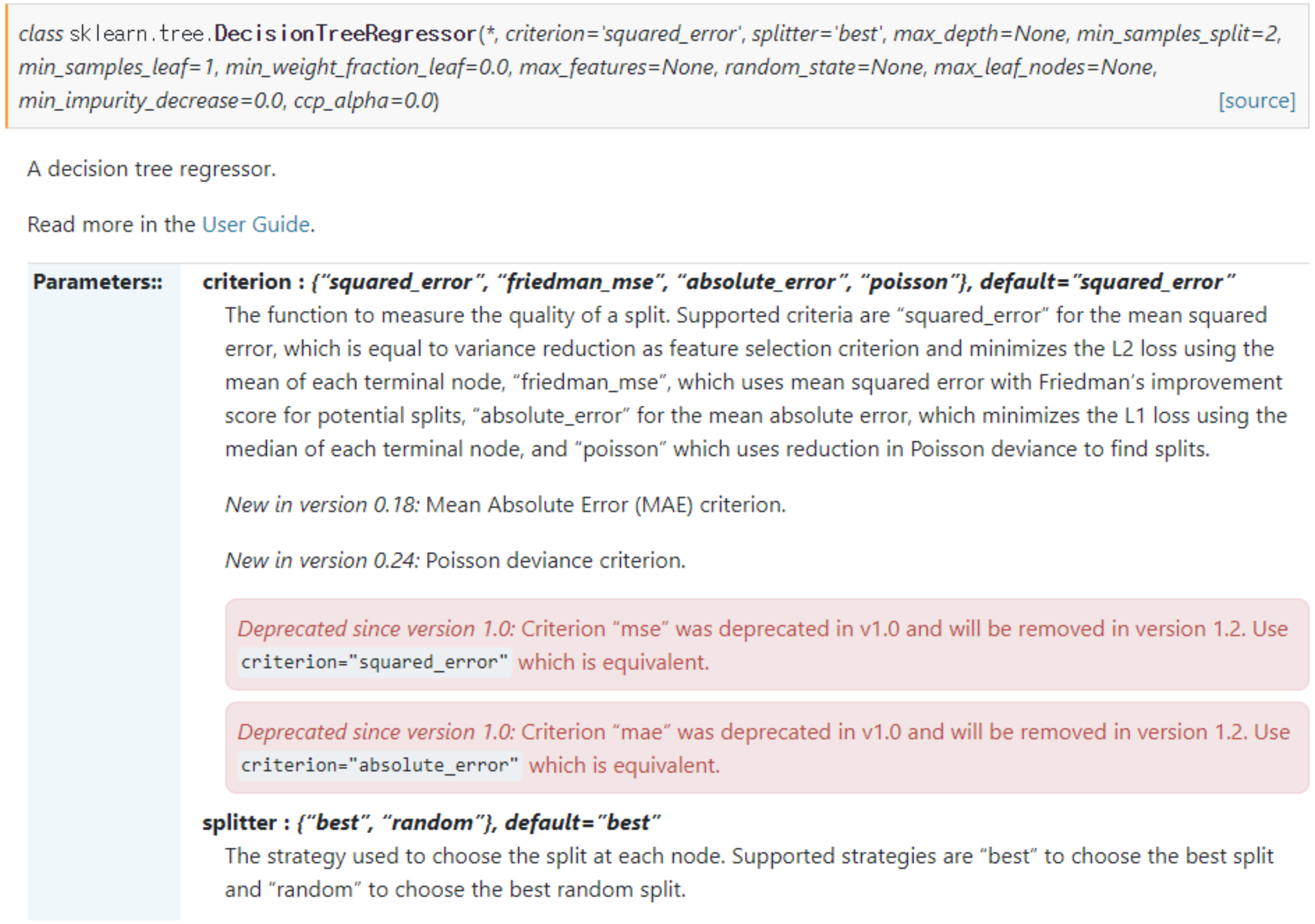
- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

1. 결정 트리

◦ 결정 트리 회귀



class sklearn.tree.DecisionTreeRegressor(**, criterion='squared_error', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha=0.0*) [source]

A decision tree regressor.

Read more in the User Guide.

Parameters::

criterion : {*"squared_error", "friedman_mse", "absolute_error", "poisson"*}, *default="squared_error"*

The function to measure the quality of a split. Supported criteria are "squared_error" for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, "friedman_mse", which uses mean squared error with Friedman's improvement score for potential splits, "absolute_error" for the mean absolute error, which minimizes the L1 loss using the median of each terminal node, and "poisson" which uses reduction in Poisson deviance to find splits.

New in version 0.18: Mean Absolute Error (MAE) criterion.

New in version 0.24: Poisson deviance criterion.

Deprecated since version 1.0: Criterion "mse" was deprecated in v1.0 and will be removed in version 1.2. Use `criterion="squared_error"` which is equivalent.

Deprecated since version 1.0: Criterion "mae" was deprecated in v1.0 and will be removed in version 1.2. Use `criterion="absolute_error"` which is equivalent.

splitter : {*"best", "random"*}, *default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

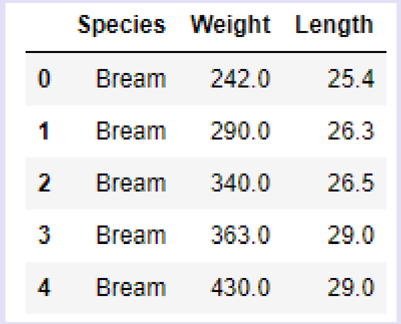
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

2. 물고기 분류 실습

- 데이터셋 구축

```
import pandas as pd

df = pd.read_csv('fish.csv')
df.head()
```

[실행결과]



	Species	Weight	Length
0	Bream	242.0	25.4
1	Bream	290.0	26.3
2	Bream	340.0	26.5
3	Bream	363.0	29.0
4	Bream	430.0	29.0

- 데이터 비율 확인

```
df['Species'].value_counts()
```

[실행결과]

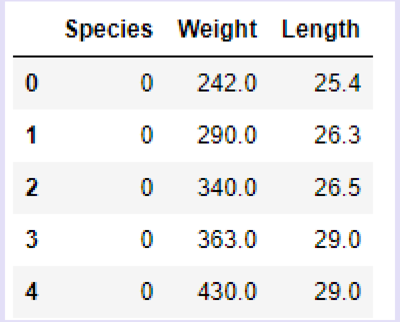
```
Bream     35
Smelt     14
Name: Species, dtype: int64
```

2. 물고기 분류 실습

- 타겟 라벨링

```
df.loc[df['Species']=='Bream','Species'] = 0
df.loc[df['Species']=='Smelt','Species'] = 1
df['Species'] = df['Species'].astype('int32')
df.head()
```

[실행결과]



```
   Species  Weight  Length
0        0   242.0    25.4
1        0   290.0    26.3
2        0   340.0    26.5
3        0   363.0    29.0
4        0   430.0    29.0
```

- 학습 특징과 타겟 분리

```
features = df[['Weight','Length']]
outcome = df['Species']
```

2. 물고기 분류 실습

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_featues, train_target, test_target = train_test_split(features,outcome,
random_state=42, stratify=outcome)
```

- 훈련 데이터 타겟 분포 확인

```
train_target.value_counts()
```

```
[실행결과]
0     26
1     10
Name: Species, dtype: int64
```

- 테스트 데이터 타겟 분포 확인

```
test_target.value_counts()
```

```
[실행결과]
0     9
1     4
Name: Species, dtype: int64
```

IMDL Intelligent Medical Data Lab

2. 물고기 분류 실습

◦ 결정 트리로 학습

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_features, train_target)
```

◦ 성능 평가

```
pred = dt.predict(test_featues)
from sklearn.metrics import accuracy_score, precision_score, recall_score

print(accuracy_score(test_target, pred))
print(precision_score(test_target, pred))
print(recall_score(test_target, pred))
```

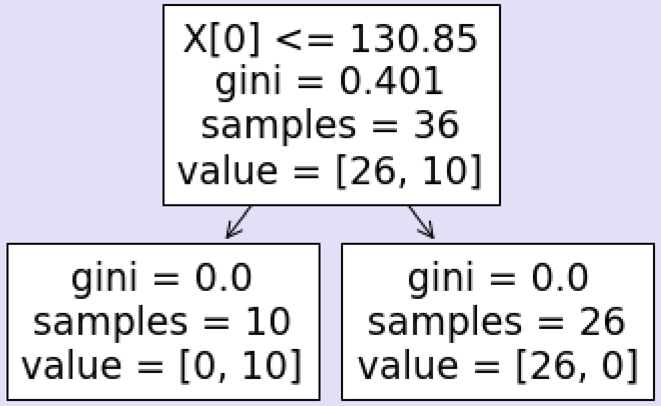
```
[실행결과]
1.0
1.0
1.0
```

2. 물고기 분류 실습

◦ 결정 트리 시각화

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plot_tree(dt)
plt.show()
```

[실행결과]

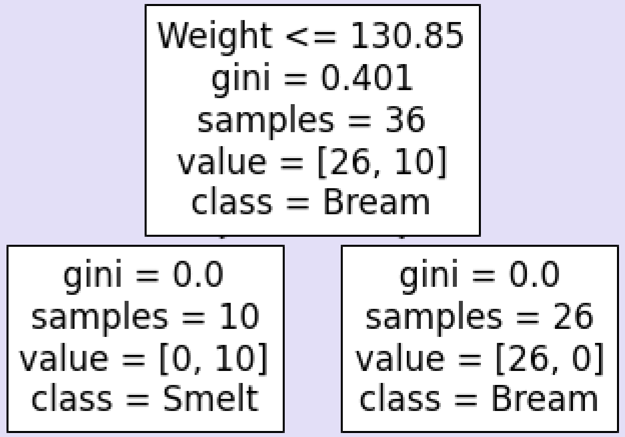


2. 물고기 분류 실습

◦ 결정 트리 시각화 개선

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plot_tree(dt,feature_names=['Weight','Height'],class_names=['Bream','Smelt'])
plt.show()
```

[실행결과]

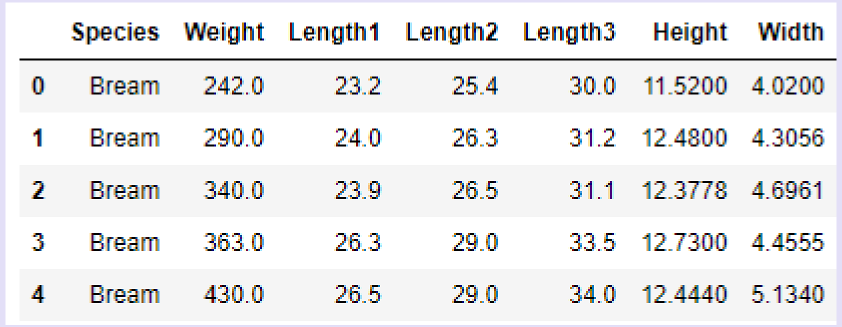


3. 더 많은 물고기 분류 실습

- 데이터셋 구축

```
import pandas as pd

df = pd.read_csv('fish_all.csv')
df.head()
```

[실행결과]



```
  Species  Weight  Length1  Length2  Length3   Height   Width
0   Bream   242.0     23.2     25.4     30.0  11.5200  4.0200
1   Bream   290.0     24.0     26.3     31.2  12.4800  4.3056
2   Bream   340.0     23.9     26.5     31.1  12.3778  4.6961
3   Bream   363.0     26.3     29.0     33.5  12.7300  4.4555
4   Bream   430.0     26.5     29.0     34.0  12.4440  5.1340
```

3. 더 많은 물고기 분류 실습

- 타겟 데이터 비율 확인

```
df['Species'].value_counts()
```

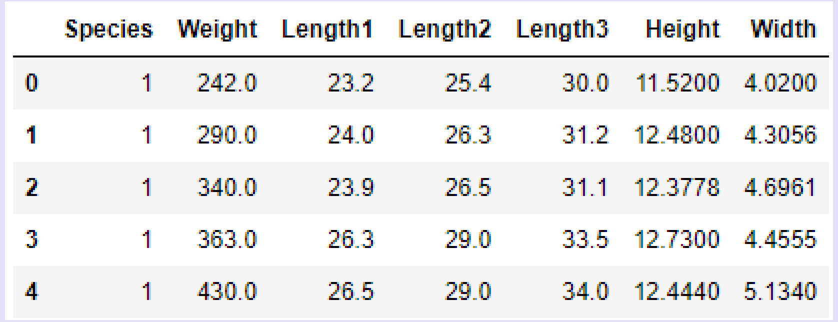
```
[실행결과]
Perch        56
Bream        35
Roach        20
Pike         17
Smelt        14
Parkki       11
Whitefish     6
Name: Species, dtype: int64
```

3. 더 많은 물고기 분류 실습

◦ 타겟 라벨링

```
df.loc[df['Species']=='Perch','Species'] = 0
df.loc[df['Species']=='Bream','Species'] = 1
df.loc[df['Species']=='Roach','Species'] = 2
df.loc[df['Species']=='Pike','Species'] = 3
df.loc[df['Species']=='Smelt','Species'] = 4
df.loc[df['Species']=='Parkki','Species'] = 5
df.loc[df['Species']=='Whitefish','Species'] = 6
df['Species'] = df['Species'].astype('int32')
df.head()
```

[실행결과]



	Species	Weight	Length1	Length2	Length3	Height	Width
0	1	242.0	23.2	25.4	30.0	11.5200	4.0200
1	1	290.0	24.0	26.3	31.2	12.4800	4.3056
2	1	340.0	23.9	26.5	31.1	12.3778	4.6961
3	1	363.0	26.3	29.0	33.5	12.7300	4.4555
4	1	430.0	26.5	29.0	34.0	12.4440	5.1340

3. 더 많은 물고기 분류 실습

- 학습 특징과 타겟 분류

```
features = df[['Weight','Length1','Length2','Length3','Height','Width']]
outcome = df['Species']
```

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_featues, train_target, test_target = train_test_split(features, outcome,
random_state=42, stratify=outcome)
```

- 결정 트리 학습

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_features, train_target)
```

3. 더 많은 물고기 분류 실습

◦ 성능 평가

```
pred = dt.predict(test_featues)

from sklearn.metrics import accuracy_score

print(accuracy_score(test_target, pred))
```

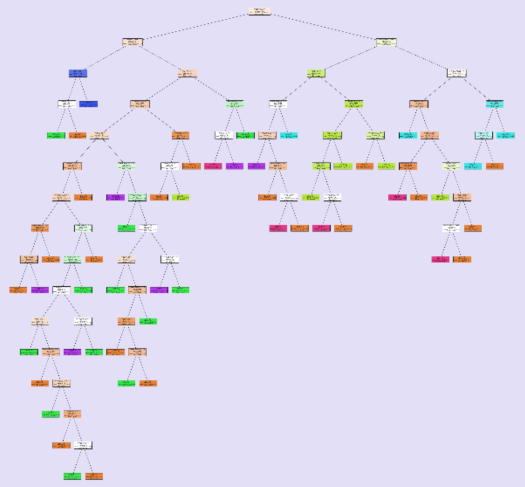
```
[실행결과]
0.725
```

3. 더 많은 물고기 분류 실습

◦ 결정 트리 시각화

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(50,50))
plot_tree(dt,feature_names=['Weight','Vertical
length'],class_names=['Perch','Bream','Roach','Pike','Smelt','Parkki','Whitefish'],filled = True)
plt.show()
```

[실행결과]

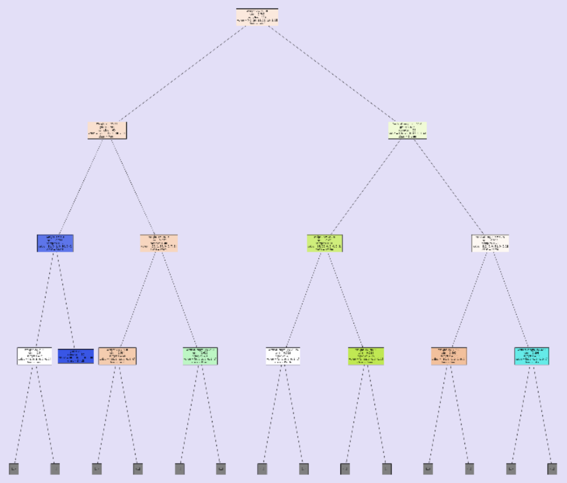


3. 더 많은 물고기 분류 실습

◦ 결정 트리 시각화 - 최대 깊이 제한

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(50,50))
plot_tree(dt,max_depth=3,feature_names=['Weight','Vertical length'],class_names=['Perch','Bream','Roach','Pike','Smelt','Parkki','Whitefish'],filled = True)
plt.show()
```

[실행결과]



3. 더 많은 물고기 분류 실습

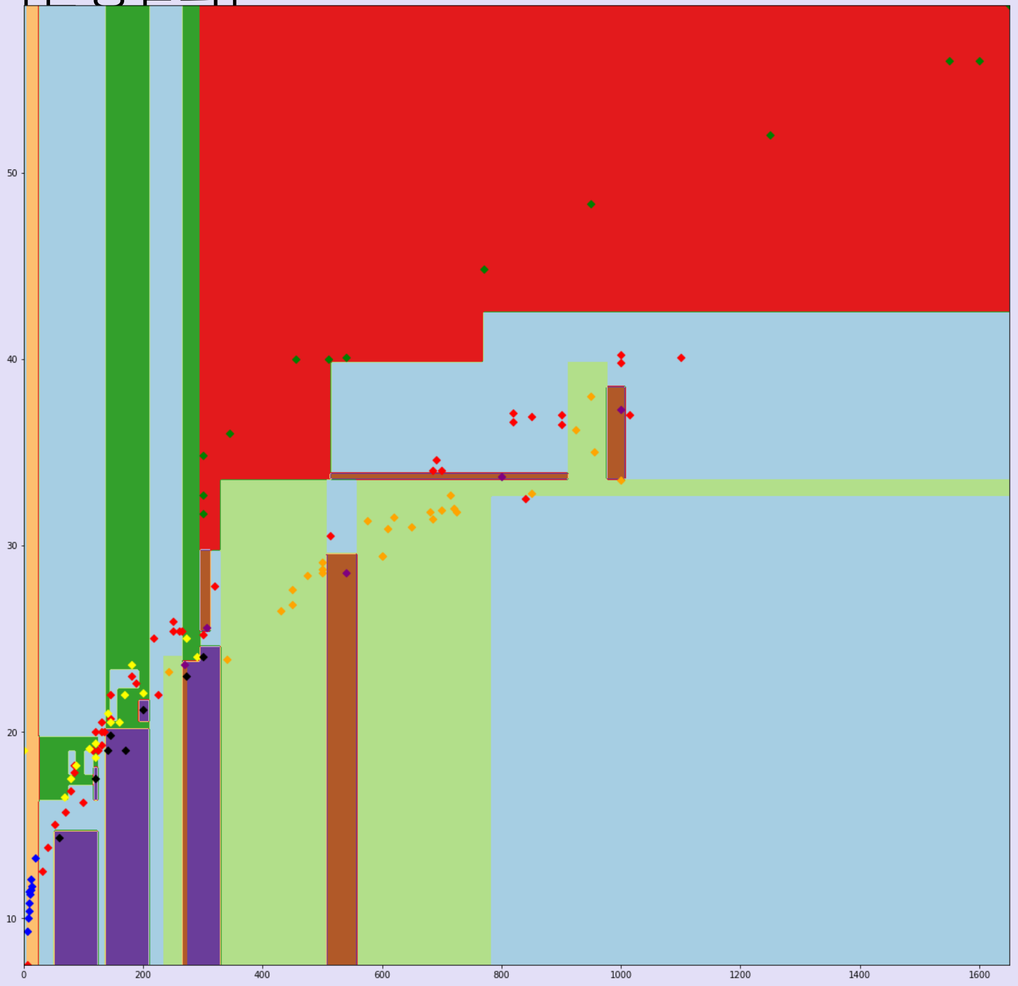
◦ 결정 트리 경계 시각화

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
color=['red','orange','yellow','green','blue','black','purple']
#      Perch  Bream    Roach    Pike   Smelt   Parkki  Whitefish
plt.figure(figsize=(20,20))
grid_size = 500
x, y = np.meshgrid(np.linspace(features['Weight'].min(), features['Weight'].max(), grid_size),
                   np.linspace(features['Length1'].min(), features['Length1'].max(), grid_size))
c = dt.predict( np.hstack([x.reshape(-1, 1), y.reshape(-1, 1)]) ).reshape(grid_size, grid_size)

plt.contourf(x, y, c, cmap='Paired')

for species in sorted(outcome.unique()) :
    feature = train_features.loc[train_target==species,['Weight','Length1']]
    plt.scatter(feature['Weight'],feature['Length1'], marker='D', color=color[species])
plt.show()
```

3. 더 많은 물고기 분류 실습

◦ 결정 트리 경계 시각화

[실행결과]



4. 결정 트리 경계 시각화 실습

- 데이터셋 구축

```
import pandas as pd

df = pd.read_csv('fish.csv')
```

- 타겟 라벨링

```
df.loc[df['Species']=='Bream','Species'] = 0
df.loc[df['Species']=='Smelt','Species'] = 1
df['Species'] = df['Species'].astype('int32')
```

- 학습 특징과 타겟 분리

```
features = df[['Weight','Length']]
outcome = df['Species']
```

4. 결정 트리 경계 시각화 실습

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_featues, train_target, test_target = train_test_split(features, outcome,
random_state=42, stratify=outcome)
```

- 결정 트리 학습

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_features, train_target)
```

IMDL
Intelligent Medical Data Lab

4. 결정 트리 경계 시각화 실습

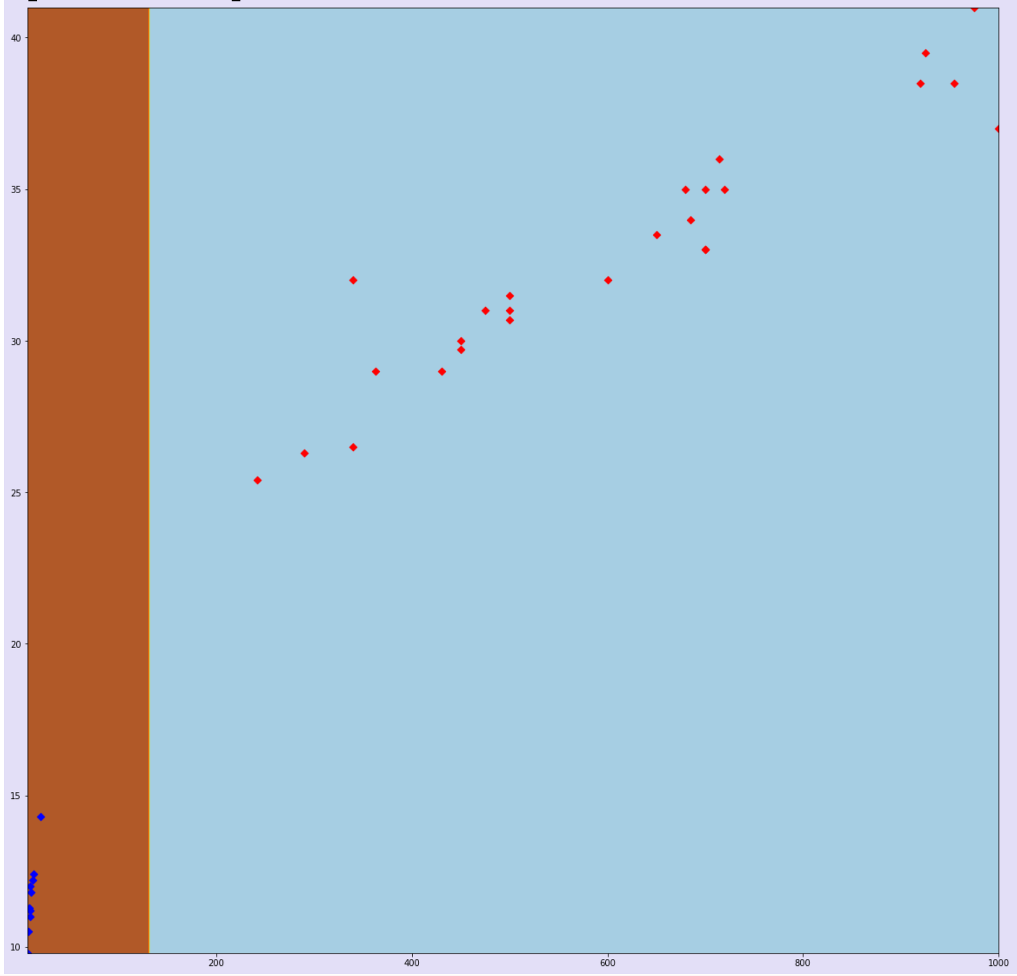
◦ 경계 시각화

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
color=['red','blue']
#       Bream Smelt

plt.figure(figsize=(20,20))
grid_size = 500
x, y = np.meshgrid(np.linspace(df['Weight'].min(), df['Weight'].max(), grid_size),
                   np.linspace(df['Length'].min(), df['Length'].max(), grid_size))
c = dt.predict( np.hstack([x.reshape(-1, 1), y.reshape(-1, 1)]) ).reshape(grid_size, grid_size)

plt.contourf(x, y, c, cmap='Paired')

for species in sorted(outcome.unique()) :
    feature = train_features.loc[outcome==species,['Weight','Length']]
    plt.scatter(feature['Weight'],feature['Length'], marker='D', color=color[species])
plt.show()
```

4. 결정 트리 경계 시각화 실습

◦ 경계 시각화

[실행결과]

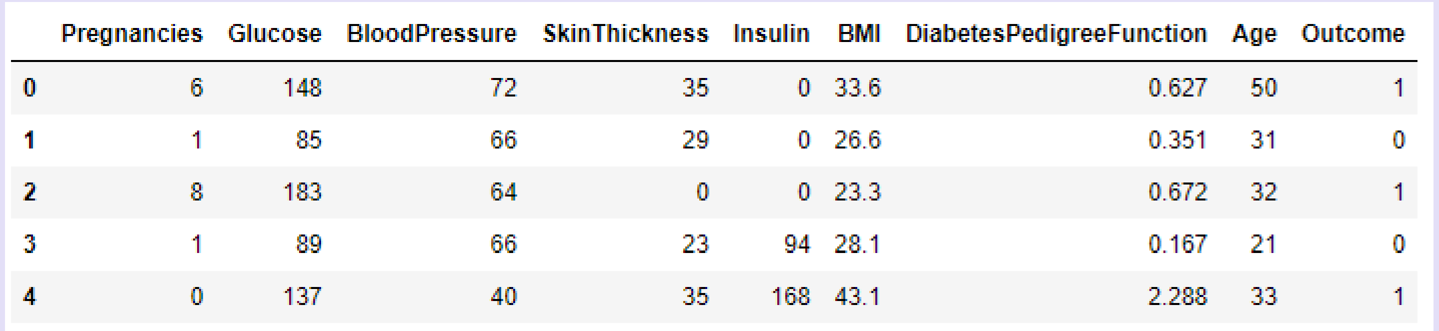


5. 당뇨병 예측 실습

◦ 데이터셋 구축

```
import pandas as pd

df = pd.read_csv('Pima_Indians_Diabetes_Database.csv')
df.head()
```

[실행결과]



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

5. 당뇨병 예측 실습

- 이상치 제거

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    df.loc[df[key] <= df[key].quantile(0.05), key]=None
    df.loc[df[key] >= df[key].quantile(0.95), key]=None
```

- 학습 특징과 타겟 분리

```
features = df[df.keys().drop('Outcome')]
outcome = df['Outcome']
```

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target = train_test_split(features,
outcome, stratify=outcome, random_state=42)
```

IMDL Intelligent Medical Data Lab

5. 당뇨병 예측 실습

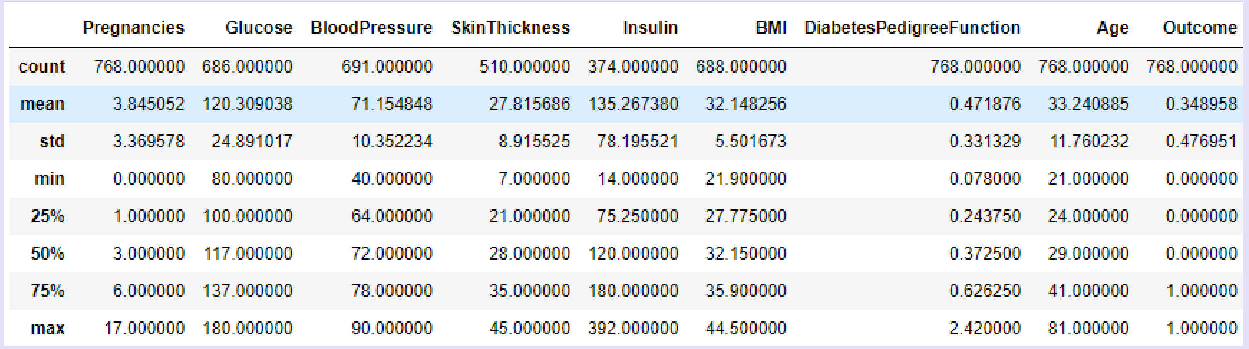
◦ 결측치 중앙값으로 보간

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    tmp = train_features[key].median()
    train_features[key].fillna(tmp,inplace=True)
    test_features[key].fillna(tmp,inplace=True)
```

◦ 데이터 분포 확인

```
df.describe()
```

[실행결과]



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	686.000000	691.000000	510.000000	374.000000	688.000000	768.000000	768.000000	768.000000
mean	3.845052	120.309038	71.154848	27.815686	135.267380	32.148256	0.471876	33.240885	0.348958
std	3.369578	24.891017	10.352234	8.915525	78.195521	5.501673	0.331329	11.760232	0.476951
min	0.000000	80.000000	40.000000	7.000000	14.000000	21.900000	0.078000	21.000000	0.000000
25%	1.000000	100.000000	64.000000	21.000000	75.250000	27.775000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	28.000000	120.000000	32.150000	0.372500	29.000000	0.000000
75%	6.000000	137.000000	78.000000	35.000000	180.000000	35.900000	0.626250	41.000000	1.000000
max	17.000000	180.000000	90.000000	45.000000	392.000000	44.500000	2.420000	81.000000	1.000000

5. 당뇨병 예측 실습

- 결정 트리 학습

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_features, train_target)
```

- 성능 평가

```
y_pred = dt.predict(test_features)

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

```
[실행결과]
Accuracy :
0.6979166666666666
Precision :
0.5584415584415584
Recall :
0.6417910447761194
F1 score :
0.5972222222222222
```

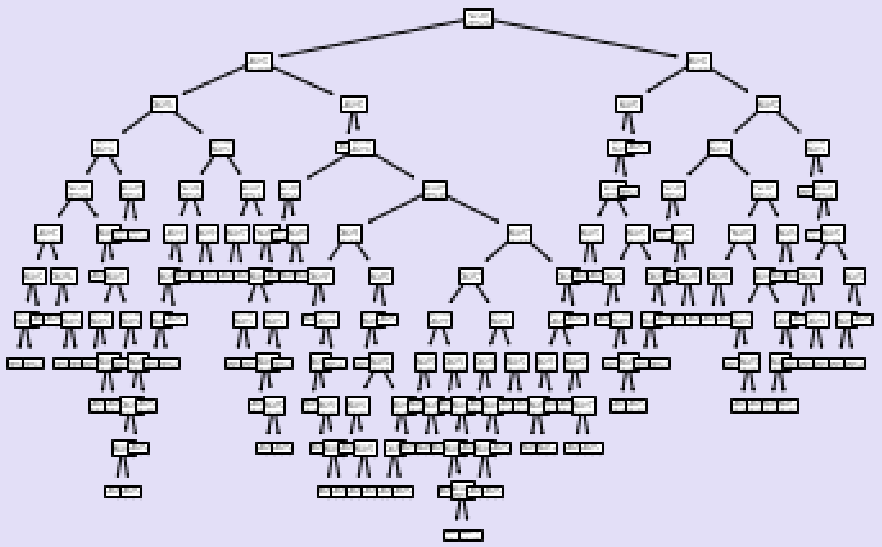
IMDL Intelligent Medical Data Lab

5. 당뇨병 예측 실습

◦ 결정 트리 시각화

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plot_tree(dt)
plt.show()
```

[실행결과]

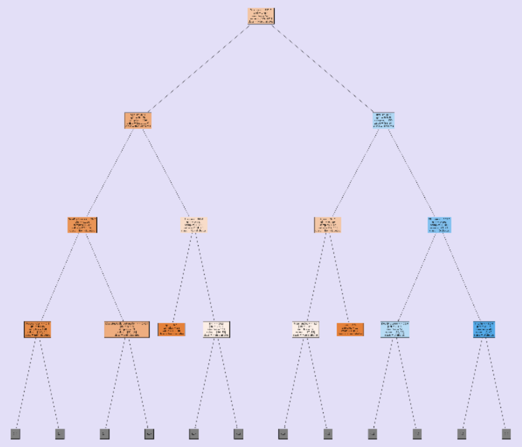


5. 당뇨병 예측 실습

◦ 결정 트리 시각화

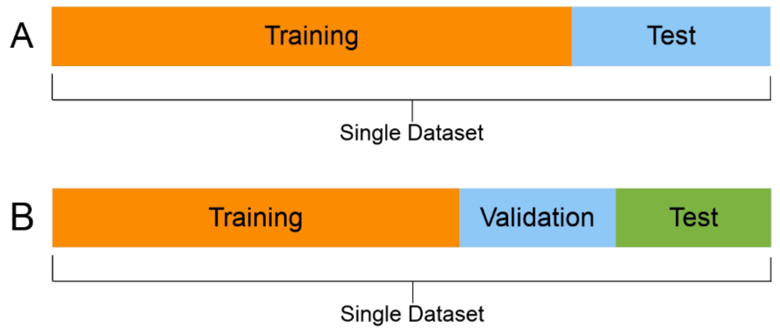
```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(50,50))
plot_tree(dt,max_depth=3,feature_names=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age'],class_names=['Non-diabetes','Diabetes'],filled = True)
plt.show()
```

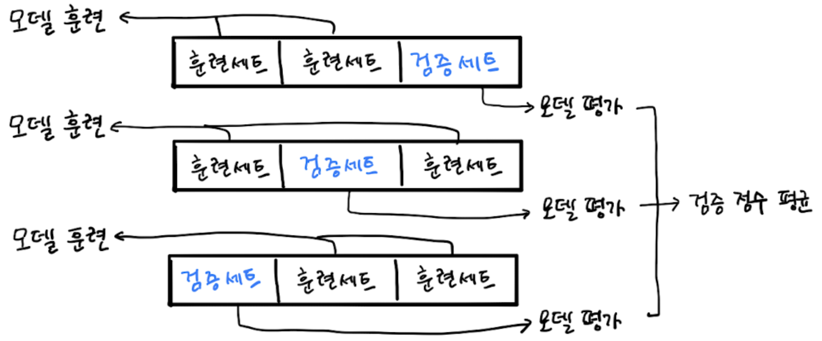
[실행결과]



6. 교차 검증과 하이퍼파라미터 튜닝

- 교차 검증
  - 훈련 데이터를 조각으로 나누어 훈련 및 검증을 수행





  - Scikit-learn 라이브러리에서 cross_val_score 를 통해 수행 가능

6. 교차 검증과 하이퍼파라미터 튜닝

◦ 하이퍼파라미터 튜닝

- 그리드 서치

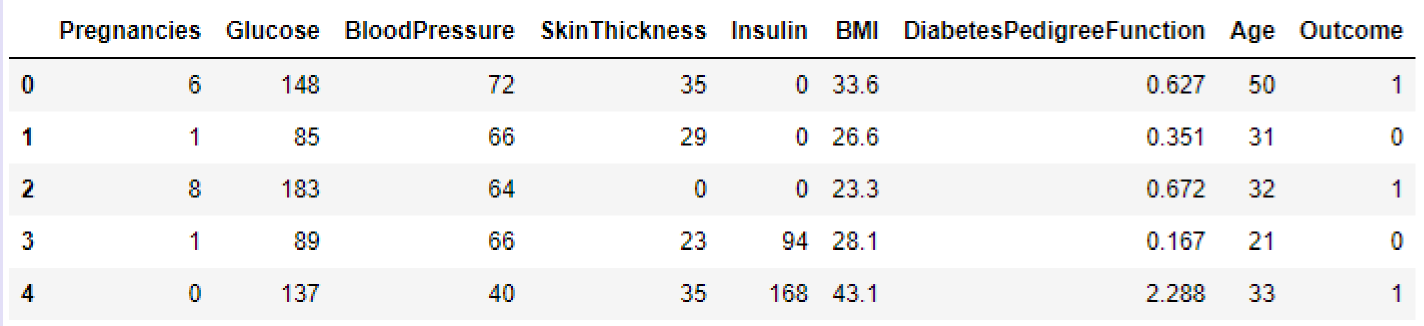
- 랜덤 서치

7. 교차 검증 실습

◦ 데이터셋 구축

```
import pandas as pd

df = pd.read_csv('Pima_Indians_Diabetes_Database.csv')
df.head()
```

[실행결과]



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

7. 교차 검증 실습

- 이상치 제거

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    df.loc[df[key] <= df[key].quantile(0.05), key]=None
    df.loc[df[key] >= df[key].quantile(0.95), key]=None
```

- 학습 특징과 타겟 분리

```
features = df[df.keys().drop('Outcome')]
outcome = df['Outcome']
```

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target = train_test_split(features,
outcome, stratify=outcome, random_state=42)
```

IMDL Intelligent Medical Data Lab

7. 교차 검증 실습

◦ 결측치 중앙값으로 보간

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    tmp = train_features[key].median()
    train_features[key].fillna(tmp,inplace=True)
    test_features[key].fillna(tmp,inplace=True)
```

◦ 결정 트리 모델 생성

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
```

7. 교차 검증 실습

- 교차 검증 수행

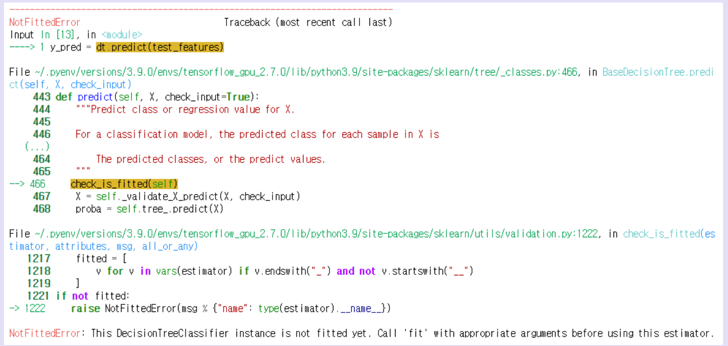
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(dt, train_features, train_target, cv=5)
print(scores)
```

```
[실행결과]
[0.61206897 0.67826087 0.67826087 0.64347826 0.71304348]
```

- 교차 검증 후 예측

```
y_pred = dt.predict(test_features)
```

[실행결과]



```
---------------------------------------------------------------------------
NotFittedError                            Traceback (most recent call last)
Input In [13], in <module>
----> 1 y_pred = dt.predict(test_features)

File ~/.pyenv/versions/3.9.0/envs/tensorflow_gpu_2.7.0/lib/python3.9/site-packages/sklearn/tree/_classes.py:466, in BaseDecisionTree.predict(self, X, check_input)
    443 def predict(self, X, check_input=True):
    444     """Predict class or regression value for X.
    445 
    446     For a classification model, the predicted class for each sample in X is
   (...)
    464         The predicted classes, or the predict values.
    465     """
--> 466     check_is_fitted(self)
    467     X = self._validate_X_predict(X, check_input)
    468     proba = self.tree_.predict(X)

File ~/.pyenv/versions/3.9.0/envs/tensorflow_gpu_2.7.0/lib/python3.9/site-packages/sklearn/utils/validation.py:1222, in check_is_fitted(estimator, attributes, msg, all_or_any)
   1217     fitted = [
   1218         v for v in vars(estimator) if v.endswith("_") and not v.startswith("__")
   1219     ]
   1221 if not fitted:
-> 1222     raise NotFittedError(msg % {"name": type(estimator).__name__})

NotFittedError: This DecisionTreeClassifier instance is not fitted yet. Call 'fit' with appropriate arguments before using this estimator.
```

7. 교차 검증 실습

◦ 교차 검증 시 인덱싱을 하므로 넘파이로 변환

```
train_features = train_features.values
train_target = train_target.values.reshape(-1,1)
```

7. 교차 검증 실습

◦ 교차 검증 하면서 학습하도록 수정

```
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score,recall_score,f1_score

stratifiedkfold = StratifiedKFold(n_splits=10, random_state=42, shuffle=True)
dt = DecisionTreeClassifier(random_state=42)

accuracy_scores = []
precision_scores = []
recall_scores = []
f1_scores = []
for train_index, test_index in stratifiedkfold.split(train_features, train_target):
    train_feature_fold = train_features[train_index]
    train_target_fold = train_target[train_index]

    val_feature_fold = train_features[test_index]
    val_target_fold = train_target[test_index]

    dt.fit(train_feature_fold, train_target_fold)

    y_pred = dt.predict(val_feature_fold)

    accuracy_scores.append(accuracy_score(val_target_fold,y_pred))
    precision_scores.append(precision_score(val_target_fold,y_pred))
    recall_scores.append(recall_score(val_target_fold,y_pred))
    f1_scores.append(f1_score(val_target_fold,y_pred))
print(accuracy_scores)
print(precision_scores)
print(recall_scores)
print(f1_scores)
```

[실행결과]

```
[0.6206896551724138,
0.7413793103448276,
0.5517241379310345,
…
```

7. 교차 검증 실습

◦ 교차 검증를 통한 학습 후 성능 평가

```
y_pred = dt.predict(test_features)

from sklearn.metrics import accuracy_score, precision_score,recall_score,f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 0.65625
Precision : 0.5076923076923077
Recall : 0.4925373134328358
F1 score : 0.5
```

7. 교차 검증 실습

◦ 혼동 행렬 출력

```
from sklearn.metrics import confusion_matrix

confusion_matrix(test_target, y_pred)
```

```
[실행결과]
array([[93, 32],
       [34, 33]])
```

◦ 혼동 행렬 라벨 순서 변경

```
from sklearn.metrics import confusion_matrix

confusion_matrix(test_target, y_pred, labels=[1,0])
```

```
[실행결과]
array([[33, 34],
       [32, 93]])
```

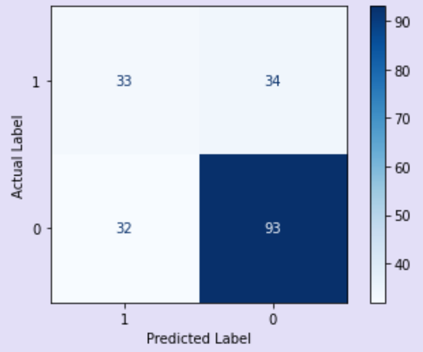
IMDL Intelligent Medical Data Lab

7. 교차 검증 실습

◦ 혼동 행렬 시각화 출력

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix

matrix = plot_confusion_matrix(dt, test_features, test_target, labels=[1,0], cmap=plt.cm.Blues)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
```

[실행결과]

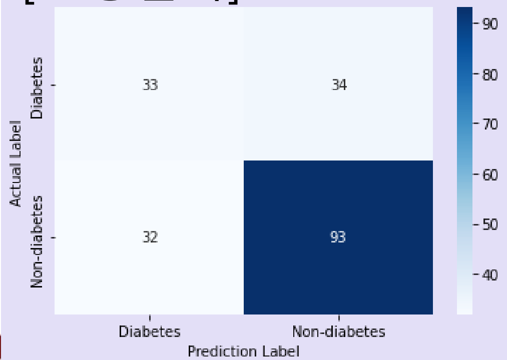


7. 교차 검증 실습

◦ seaborn 라이브러리를 통한 출력

```
%matplotlib inline

from sklearn.metrics import confusion_matrix
import seaborn as sns

c_matrix = confusion_matrix(test_target, y_pred, labels=[1,0])
ax = sns.heatmap(c_matrix, annot = True, fmt='d', xticklabels=['Diabetes', 'Non-diabetes'],
yticklabels=['Diabetes', 'Non-diabetes'], cmap='Blues')
ax.set_xlabel("Prediction Label")
ax.set_ylabel("Actual Label")
plt.show()
```

[실행결과]

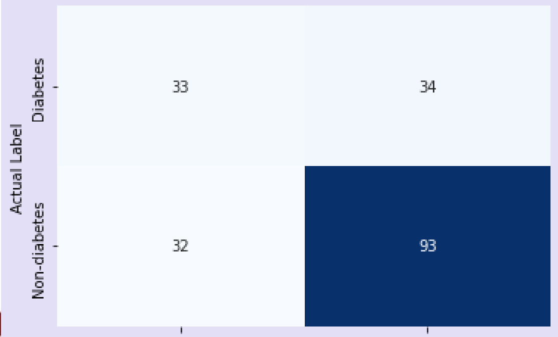


7. 교차 검증 실습

◦ seaborn 라이브러리를 통한 출력2

```
%matplotlib inline

from sklearn.metrics import confusion_matrix
import seaborn as sns

c_matrix = confusion_matrix(test_target, y_pred, labels=[1,0])
ax = sns.heatmap(c_matrix, annot = True, fmt='d', xticklabels=['Diabetes', 'Non-diabetes'],
yticklabels=['Diabetes', 'Non-diabetes'], cbar=False, cmap='Blues')
ax.set_xlabel("Prediction Label")
ax.set_ylabel("Actual Label")
plt.show()
```

[실행결과]



8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 하이퍼파라미터 튜닝 전 결정 트리 학습

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_features, train_target)
```

◦ 성능 평가

```
y_pred = dt.predict(test_features)
from sklearn.metrics import accuracy_score,
precision_score
from sklearn.metrics import recall_score,f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

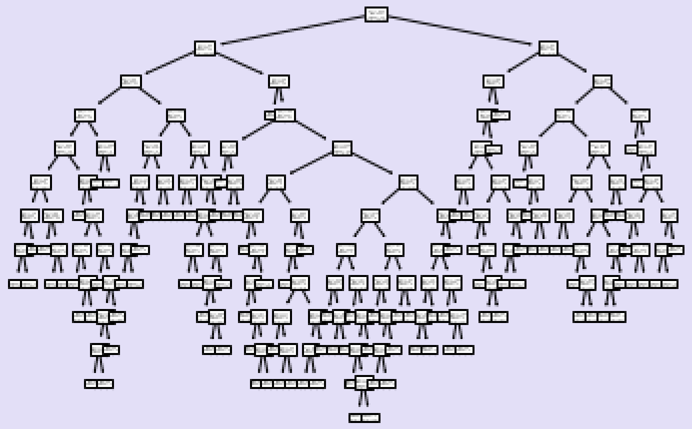
```
[실행결과]
Accuracy :
0.6979166666666666
Precision :
0.5584415584415584
Recall :
0.6417910447761194
F1 score :
0.5972222222222222
```

8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 결정 트리 시각화

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plot_tree(dt)
plt.show()
```

[실행결과]

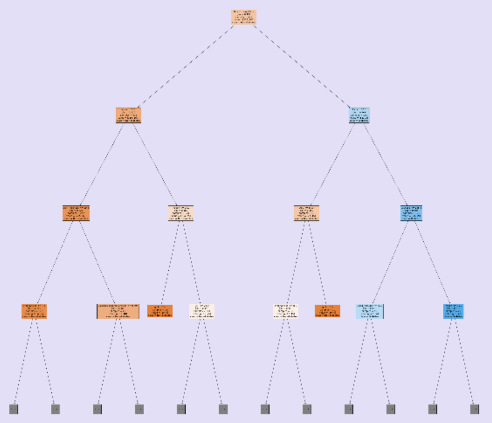


8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 결정 트리 시각화

```
%matplotlib inline
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(50,50))
plot_tree(dt,max_depth=3,feature_names=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age'],class_names=['Non-diabetes','Diabetes'],filled = True)
plt.show()
```

[실행결과]



8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 그리드 서치를 통한 하이퍼파라미터 튜닝 설정

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
import numpy as np

dt = DecisionTreeClassifier(random_state=42)
parameters = {'criterion':['gini','entropy'], 'max_depth':np.arange(1,11),
'min_samples_split':np.arange(2,11), 'max_features':['auto','sqrt','log2',None]}
grid_search = GridSearchCV(dt, param_grid=parameters, cv=10, n_jobs=-1, scoring='recall')
grid_search.fit(train_features, train_target)
```

8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

- 주어진 파라미터 중 최적의 파라미터 확인

```
grid_search.best_params_
```

```
[실행결과]
{'criterion': 'entropy',
 'max_depth': 4,
 'max_features': None,
 'min_samples_split': 2}
```

- 하이퍼파라미터 결과 성능 평가

```
grid_search.best_score_
```

```
[실행결과]
0.6730952380952381
```

IMDL
Intelligent Medical Data Lab

8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 최적의 성능을 보이는 학습기 접근

```
dt = grid_search.best_estimator_
```

◦ 테스트 데이터로 성능 평가

```
y_pred = dt.predict(test_features)
from sklearn.metrics import accuracy_score, precision_score,recall_score,f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 0.6822916666666666
Precision : 0.5384615384615384
Recall : 0.6268656716417911
F1 score : 0.5793103448275863
```

8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 랜덤 서치를 통한 하이퍼파라미터 튜닝 설정

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
import numpy as np

dt = DecisionTreeClassifier(random_state=42)
parameters = {'criterion':['gini','entropy'],'max_depth':np.arange(1,101),
'min_samples_split':np.arange(2,101),'max_features':['auto','sqrt','log2',None]}
random_search = RandomizedSearchCV(dt, param_distributions=parameters, cv=10,
n_jobs=-1,scoring='recall',n_iter=20)
```

IMDL Intelligent Medical Data Lab

8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

- 주어진 파라미터 중 최적의 파라미터 확인

```
random_search.best_params_
```

```
[실행결과]
{'min_samples_split': 84,
 'max_features': None,
 'max_depth': 69,
 'criterion': 'entropy'}
```

- 하이퍼파라미터 결과 성능 평가

```
random_search.best_score_
```

```
[실행결과]
0.5928571428571429
```

IMDL Intelligent Medical Data Lab

8. 하이퍼파라미터 튜닝을 위한 그리드 서치와 랜덤 서치 실습

◦ 최적의 성능을 보이는 학습기 접근

```
dt = random_search.best_estimator_
```

◦ 테스트 데이터로 성능 평가

```
y_pred = dt.predict(test_features)
from sklearn.metrics import accuracy_score, precision_score,recall_score,f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 0.703125
Precision : 0.5735294117647058
Recall : 0.582089552238806
F1 score : 0.5777777777777778
```

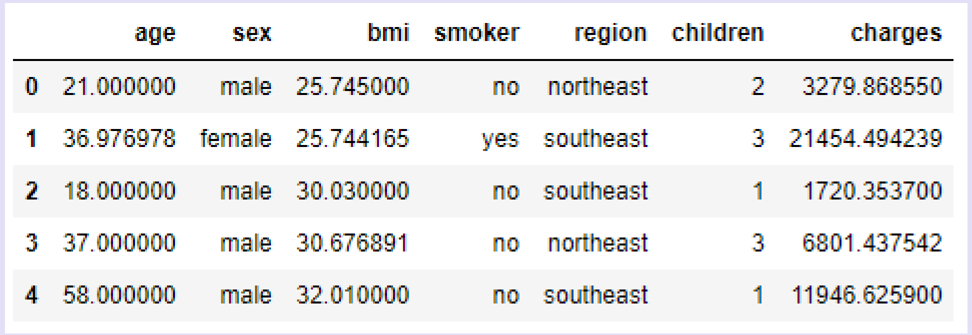
IMDL Intelligent Medical Data Lab

9. 보험료 예측

◦ 데이터셋 구축

```
import pandas as pd

df = pd.read_csv('Medical_Insurance_dataset.csv')
df.head()
```

[실행결과]



```
         age     sex        bmi  smoker     region  children       charges
0  21.000000    male  25.745000      no  northeast         2   3279.868550
1  36.976978  female  25.744165     yes  southeast         3  21454.494239
2  18.000000    male  30.030000      no  southeast         1   1720.353700
3  37.000000    male  30.676891      no  northeast         3   6801.437542
4  58.000000    male  32.010000      no  southeast         1  11946.625900
```

9. 보험료 예측

◦ 성별, 흡연 여부 라벨링 수행

```
df.loc[df['sex']=='male','sex']=0
df.loc[df['sex']=='female','sex']=1
df['sex']=df['sex'].astype('int32')

df.loc[df['smoker']=='no','smoker']=0
df.loc[df['smoker']=='yes','smoker']=1
df['smoker']=df['smoker'].astype('int32')
```

9. 보험료 예측

◦ 지역 원 핫 인코딩 수행

```
df = pd.get_dummies(df)
df.head()
```

[실행결과]

	age	sex	bmi	smoker	children	charges	region_northeast	region_northwest	region_southeast	region_southwest
0	21.000000	0	25.745000	0	2	3279.868550	1	0	0	0
1	36.976978	1	25.744165	1	3	21454.494239	0	0	1	0
2	18.000000	0	30.030000	0	1	1720.353700	0	0	1	0
3	37.000000	0	30.676891	0	3	6801.437542	1	0	0	0
4	58.000000	0	32.010000	0	1	11946.625900	0	0	1	0

IMDL Intelligent Medical Data Lab

9. 보험료 예측

◦ 훈련 특징과 타겟 나누기

```
feature = df[df.keys().drop('charges')].values
outcome = df['charges'].values
```

◦ 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

◦ 결정 트리 학습

```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(random_state=42)
dt.fit(train_input, train_target)
```

9. 보험료 예측

◦ 성능 평가

```
y_pred = dt.predict(test_input)

import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

[실행결과]

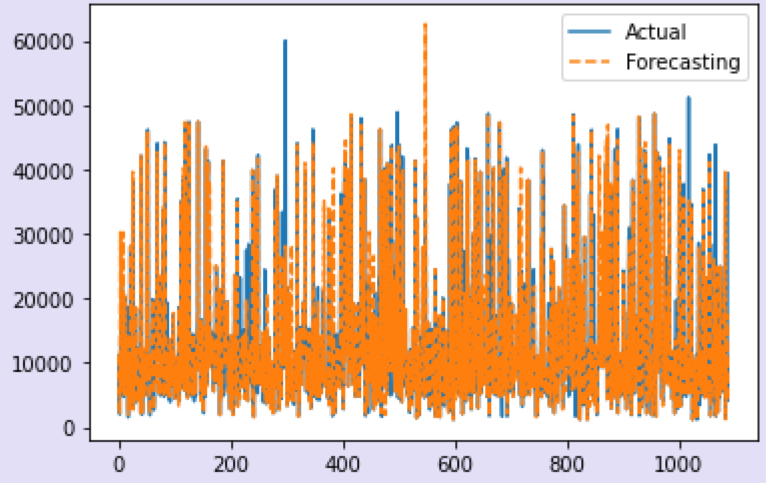
```
MAE : 1580.8816363657854
RMSE : 4752.215501095879
```

9. 보험료 예측

- 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='Actual')
plt.plot(y_pred, linestyle='--', label='Forecasting')
plt.legend()
plt.show()
```

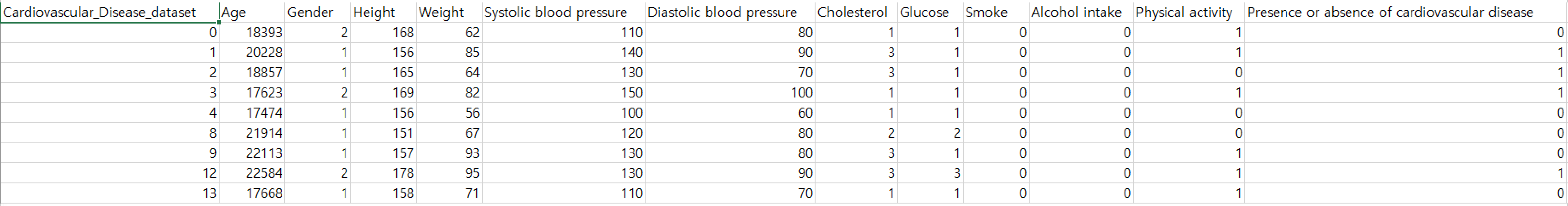
[실행결과]



10. Homework

◦ 스스로 해보기

- Cardiovascular Disease dataset 데이터셋을 이용하여 결정 트리를 이용하여 심혈관 질환 분류를 수행하여라.



Cardiovascular_Disease_dataset	Age	Gender	Height	Weight	Systolic blood pressure	Diastolic blood pressure	Cholesterol	Glucose	Smoke	Alcohol intake	Physical activity	Presence or absence of cardiovascular disease
0	18393	2	168	62	110	80	1	1	0	0	1	0
1	20228	1	156	85	140	90	3	1	0	0	1	1
2	18857	1	165	64	130	70	3	1	0	0	0	1
3	17623	2	169	82	150	100	1	1	0	0	1	1
4	17474	1	156	56	100	60	1	1	0	0	0	0
8	21914	1	151	67	120	80	2	2	0	0	0	0
9	22113	1	157	93	130	80	3	1	0	0	1	0
12	22584	2	178	95	130	90	3	3	0	0	1	1
13	17668	1	158	71	110	70	1	1	0	0	1	0

IMDL
Intelligent Medical Data Lab