Decision-Tree for Classification

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

Course Overview

- **Chap 1**: Classification And Regression Tree (CART)
- **Chap 2**: The Bias-Variance Tradeoff 편향 - 분산 트레이드 오프와 모델 앙상블의 개념
- **Chap 3**: Bagging and Random Forests
- **Chap 4**: Boosting
- **Chap 5**: Model Tuning 하이퍼파라미터 튜닝을 통해 모델을 최대한 활용하는 방법을 이해하기

Classification-tree

- Sequence of if-else questions about individual features.
- **Objective**: infer class labels.
- Able to capture non-linear relationships between features and labels.
- Don't require feature scaling (ex: Standardization, ..)

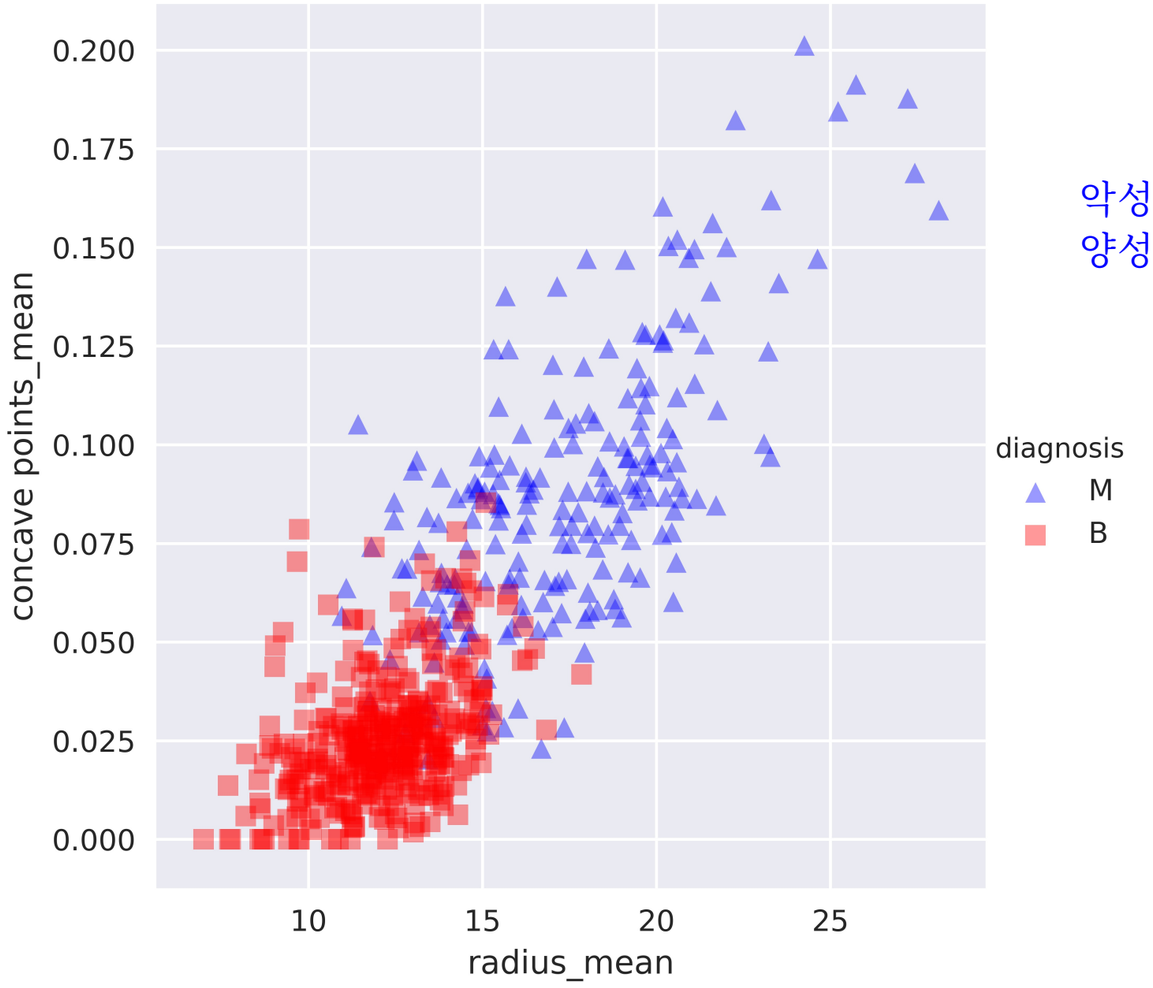
레이블이 지정된 데이터 세트가 주어지면 decision tree는 label을 추론하기 위해 개별 feature에 대한 if - else 질문을 함

linear model과 달리 트리는 feature 간의 비선형 관계를 캡쳐할 수 있음

또한, 트리는 표준화를 통해 기능이 동일한 규모 일 필요가 없음

Breast Cancer Dataset in 2D

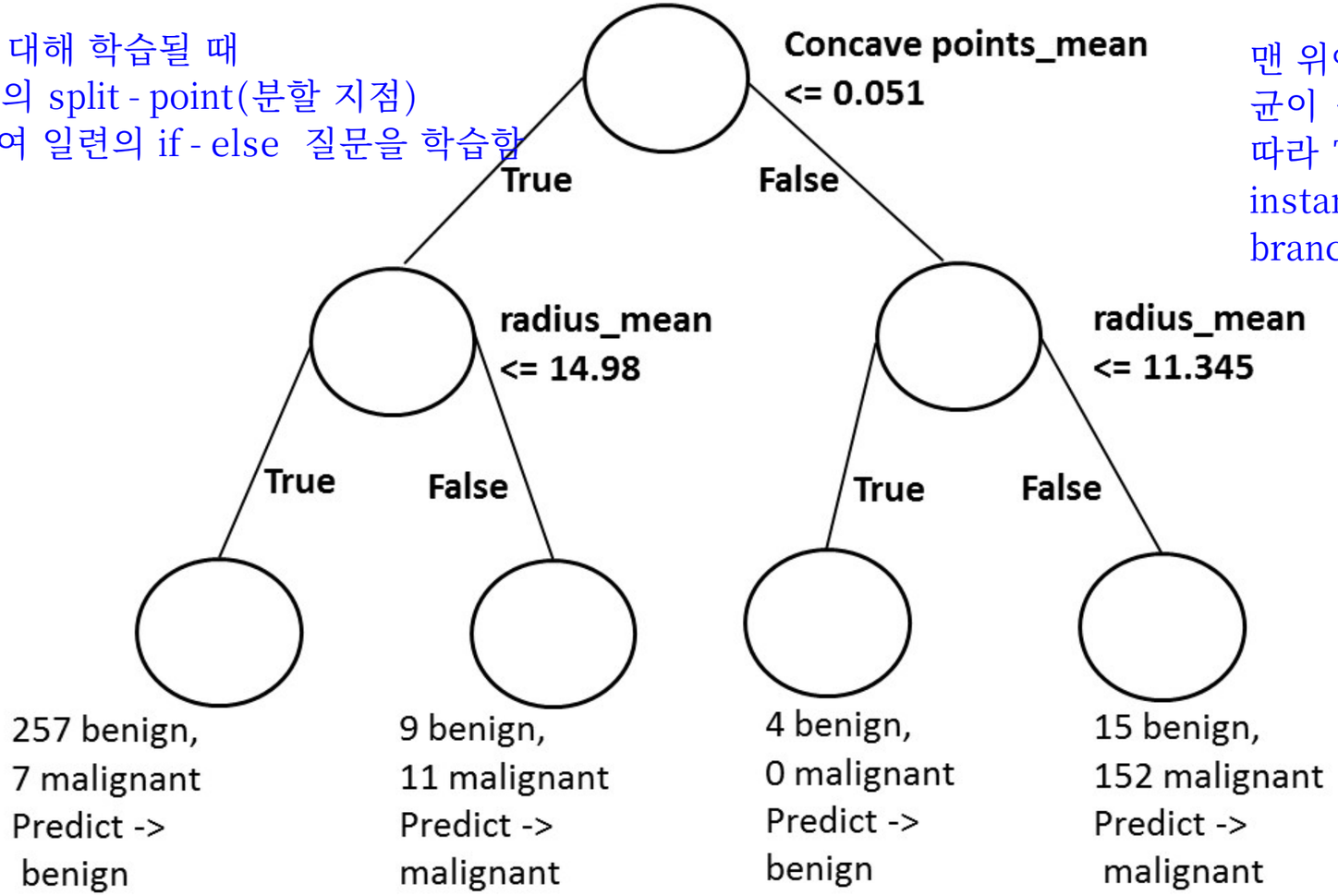
Tree를 더 구체적으로 이해하기 위해 그린 scatter



악성 종양은 blue
양성 종양은 red

Decision-tree Diagram

분류 tree가 이 데이터 세트에 대해 학습될 때
트리는 하나의 feature과 하나의 split - point(분할 지점)
을 포함하는 각 질문을 사용하여 일련의 if - else 질문을 학습함



맨 위에서 tree는 instance의 오목점 평균이 <=0.051인지 여부를 묻고 이에 따라 True, False branch를 통과함. instance는 끝에 도달할 때까지 내부 branch를 계속 순회

Classification-tree in scikit-learn

```
# Import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
# Import train_test_split
from sklearn.model_selection import train_test_split
# Import accuracy_score
from sklearn.metrics import accuracy_score
# Split the dataset into 80% train, 20% test
X_train, X_test, y_train, y_test= train_test_split(X, y,
                                                   test_size=0.2,
                                                   stratify=y,
                                                   random_state=1)

# Instantiate dt
dt = DecisionTreeClassifier(max_depth=2, random_state=1)
```

모델 성능에 대한 편향되지 않은 추정치를 얻기 위해 dataset split

max_depth 매개 변수를 2로 설정하여 최대 깊이가 2인 분류기인 DT
재현을 위한 random_state

Classification-tree in scikit-learn

```
# Fit dt to the training set
dt.fit(X_train,y_train)


# Predict the test set labels
y_pred = dt.predict(X_test)
# Evaluate the test-set accuracy
accuracy_score(y_test, y_pred)
```

```
0.9035087719298245
```

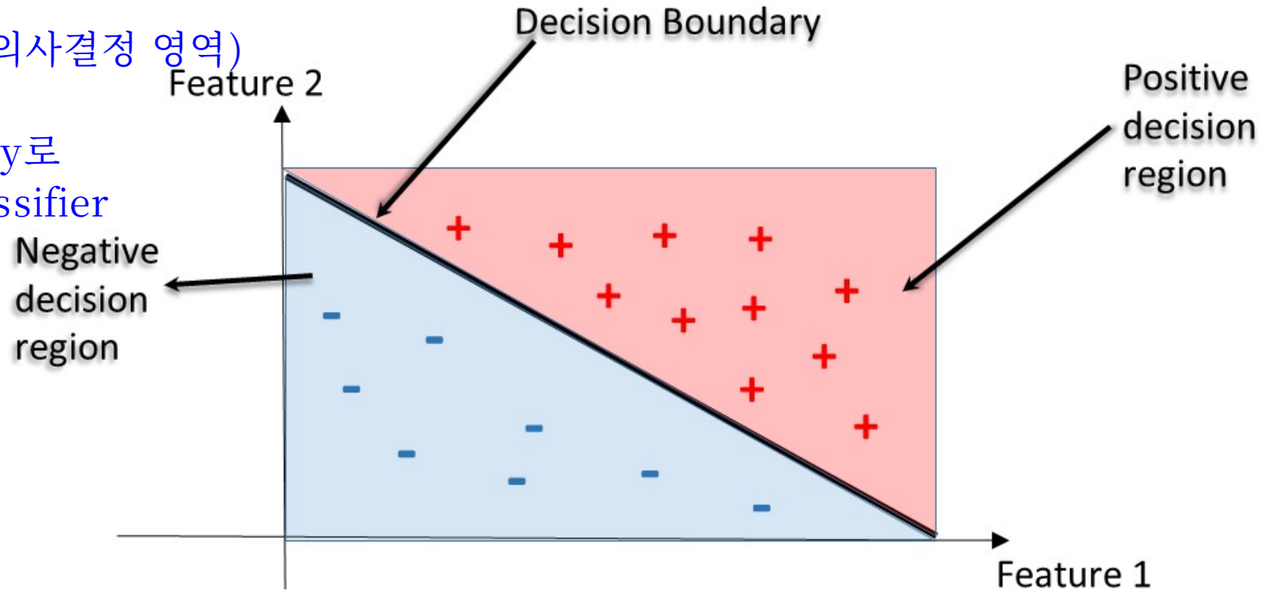
Decision Regions

Decision region: region in the feature space where all instances are assigned to one class label.

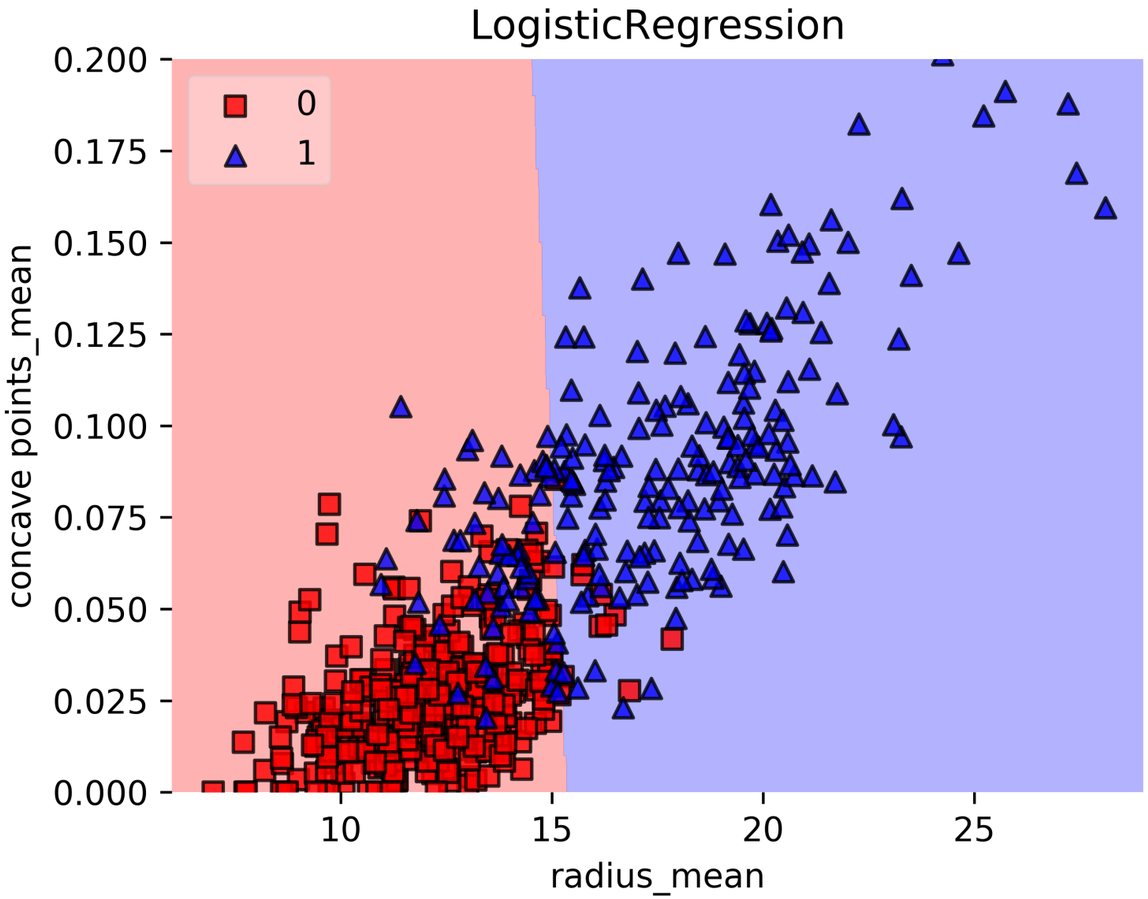
Decision Boundary: surface separating different decision regions.

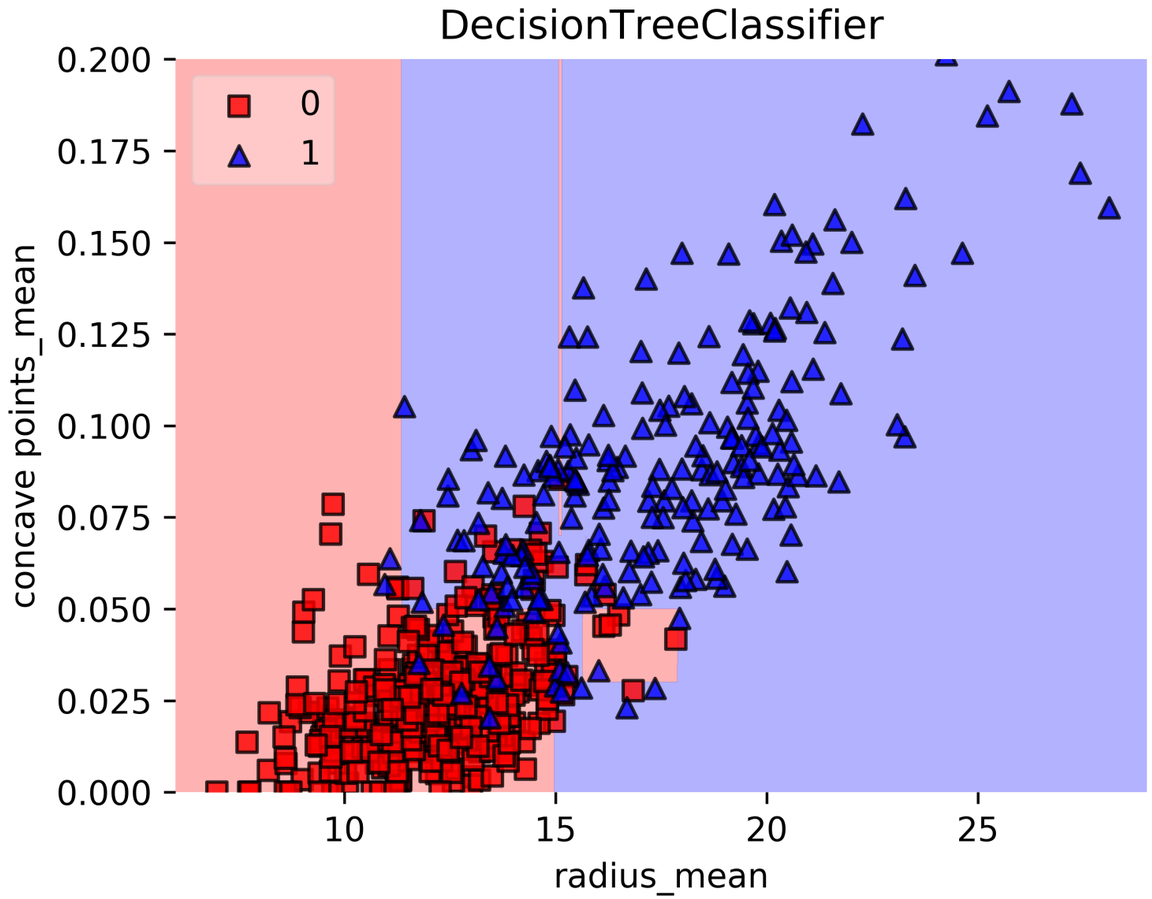
분류 모델은 feature 공간을 하나의 영역 내의
모든 인스턴스가 하나의 클래서 label에만
할당되는 영역으로 나눔
=> 이러한 영역을 decision region(의사결정 영역)

decision region은 decision boundary로
표면이 구분되고 이 그림은 linear classifier



Decision Regions: CART vs. Linear Model





DT는 feature 공간에서 직사각형 decision - region을 생성
이것은 tree에 의해 만들어진 각 분할에서 하나의 feature만 관련되기 때문에 발생

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Classification-Tree Learning

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

Building Blocks of a Decision-Tree

- **Decision-Tree**: data structure consisting of a hierarchy of nodes.
- **Node**: question or prediction.

DT는 node라고하는 개별 단위의 계층 구조로 구성된 데이터 구조
Node: 질문이나 예측을 포함하는 지점

Building Blocks of a Decision-Tree

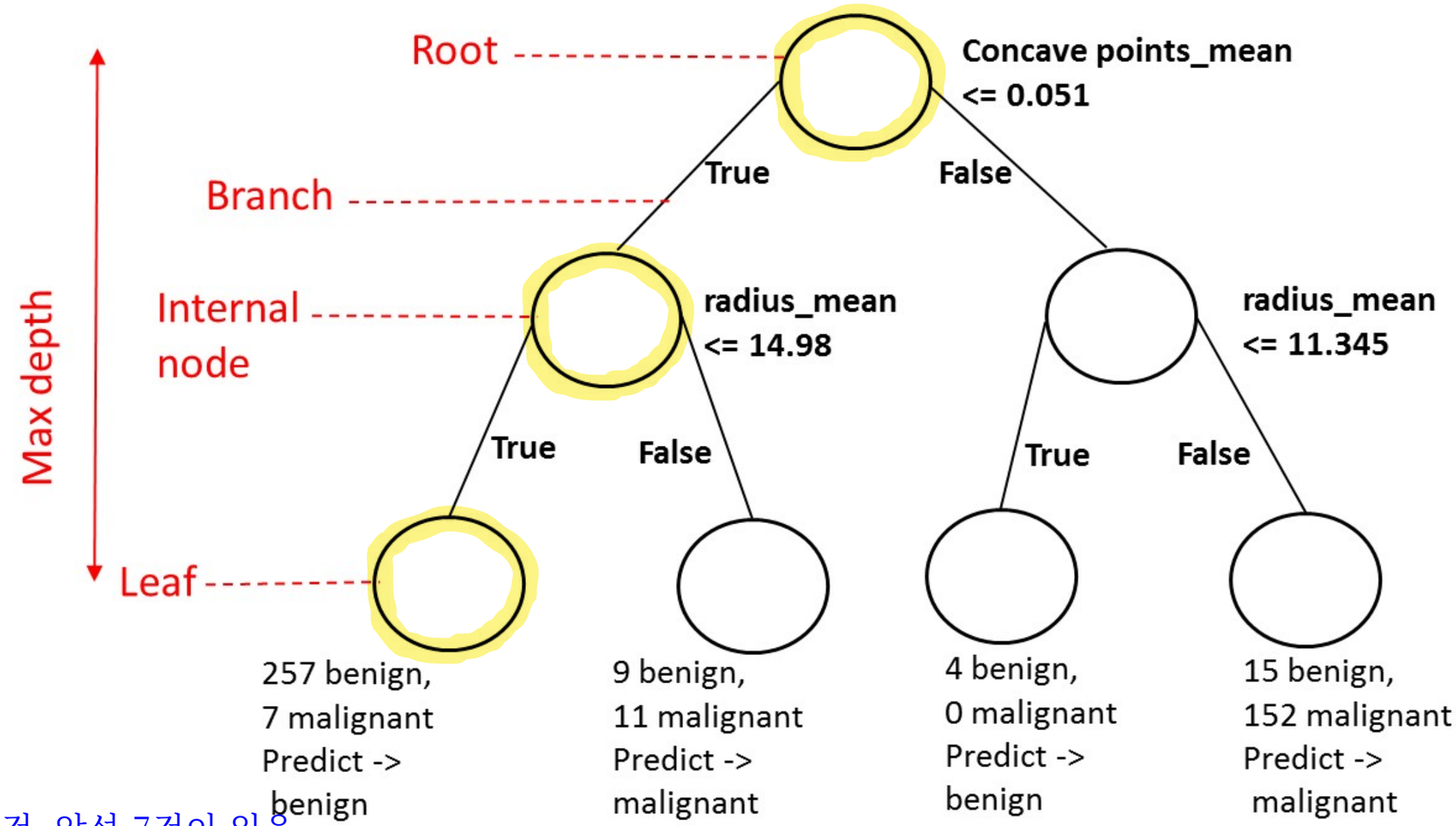
Three kinds of nodes:

- **Root**: *no* parent node, question giving rise to *two* children nodes.
- **Internal node**: *one* parent node, question giving rise to *two* children nodes.
- **Leaf**: *one* parent node, *no* children nodes --> *prediction*.

Root : 의사 결정 트리가 시작하는 node. 부모 node가 없으며 두 개의 분기를 통해 두 개의 자식 노드를 생성하는 질문이 포함
내부 노드는 부모가 있는 노드. 또한 2개의 하위 노드를 발생시키는 질문도 포함
leaf : 자식이 없는 노드. 하나의 상위 노드가 있으며 질문이 없음 = > 예측이 이루어지는 곳

분류 트리가 레이블이 지정된 데이터 세트에서 학습되면 나무는 가장 순순한 leaf를 생성하는 방식으로 feature에서 패턴을 학습

Prediction

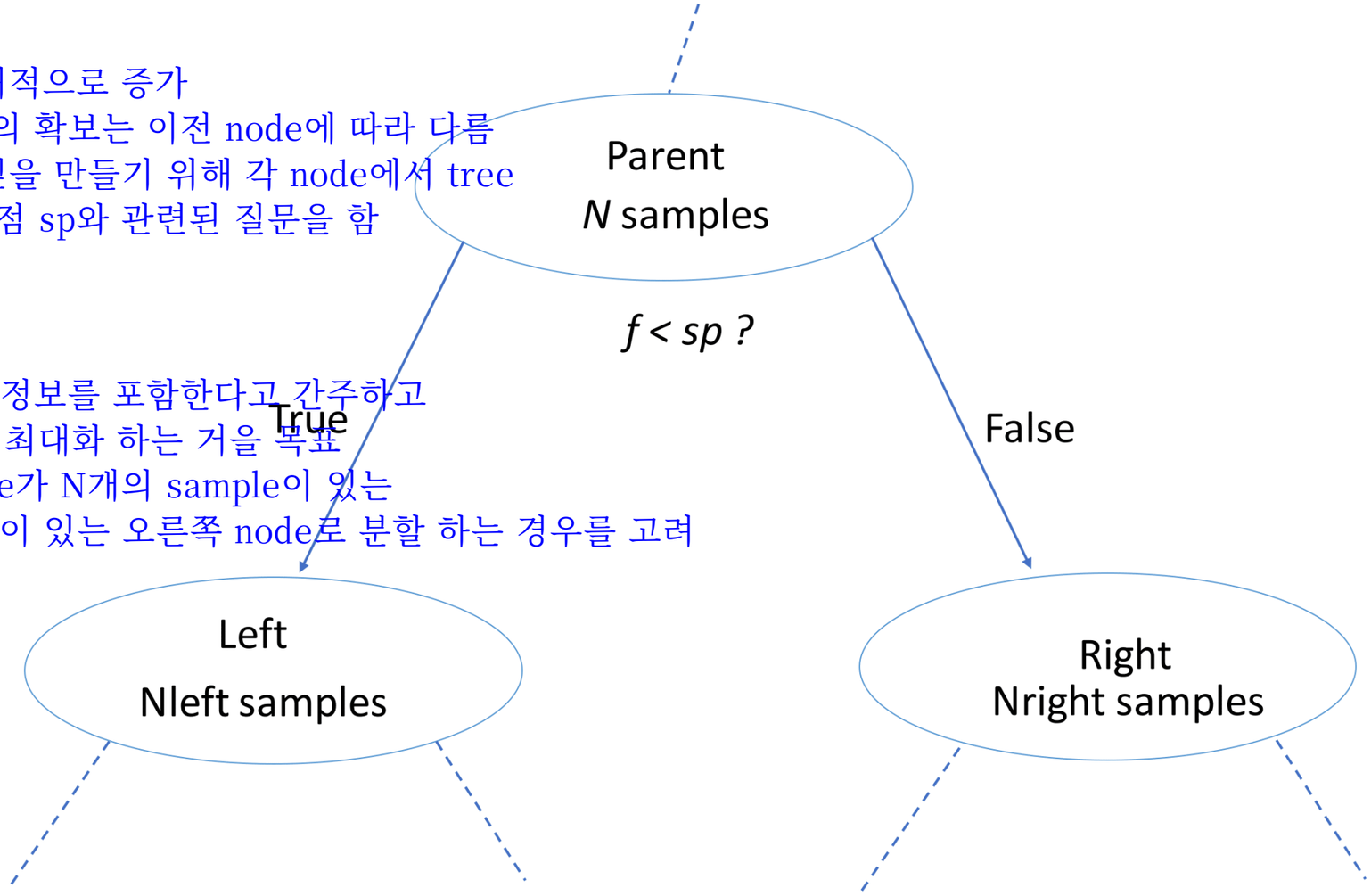


이 leaf에는 양성 257건, 악성 7건이 있음
그 결과 이 instance에 대한 tree의 예측은
'begin'이 됨

Information Gain (IG)

분류 tree의 node는 재귀적으로 증가
즉, 내부 node 또는 leaf의 확보는 이전 node에 따라 다름
가능한 한 가장 순수한 잎을 만들기 위해 각 node에서 tree
하나의 feature f와 분할점 sp와 관련된 질문을 함

이 tree는 모든 node가 정보를 포함한다고 간주하고
각 분할 후에 얻은 IG를 최대화 하는 거을 목표
N개의 샘플이 있는 node가 N개의 sample이 있는
왼쪽 node와 right 샘플이 있는 오른쪽 node로 분할 하는 경우를 고려



Information Gain (IG)

$$IG(\underbrace{f}_{feature}, \underbrace{sp}_{split-point}) = I(parent) - \left(\frac{N_{left}}{N} I(left) + \frac{N_{right}}{N} I(right) \right)$$

Criteria to measure the impurity of a node $I(node)$:

- gini index,
- entropy. ...

여기서 염두에 두어야 할 질문 : "node의 불순물을 측정하는데 사용되는 기준은 무엇인가?"
= > gini 계수와 entropy 등 다양한 기준을 사용할 수 있음

Classification-Tree Learning

- Nodes are grown recursively.
- At each node, split the data based on:
  - feature f and split-point sp to maximize $IG(\text{node})$.
- If $IG(\text{node}) = 0$, declare the node a leaf. ...

제한되지 않은 tree가 훈련되면 node가 재귀적으로 증가

non - leaf node에서는 IG를 최대화하기 위해 feature f 및 분할점 sp에 근거해 데이터를 분할

node를 분할하여 얻은 IG가 null이면 node를 leaf라고 선언 - > 이러한 규칙은 제한되지 않은 tree에 대한 것

ex) 최대 깊이를 2로 제한하면 node를 분할하여 얻은 IG가 null이 아니더라도 깊이 2의 모든 node가 leaf라고 선언됨

```
# Import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
# Import train_test_split
from sklearn.model_selection import train_test_split
# Import accuracy_score
from sklearn.metrics import accuracy_score
# Split dataset into 80% train, 20% test
X_train, X_test, y_train, y_test= train_test_split(X, y,
                                                   test_size=0.2,
                                                   stratify=y,
                                                   random_state=1)
# Instantiate dt, set 'criterion' to 'gini'
dt = DecisionTreeClassifier(criterion='gini', random_state=1)
```

Information Criterion in scikit-learn

```
# Fit dt to the training set
dt.fit(X_train,y_train)

# Predict test-set labels
y_pred= dt.predict(X_test)

# Evaluate test-set accuracy
accuracy_score(y_test, y_pred)
```

```
0.9210526315789473
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Decision-Tree for Regression

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

회귀 분석에서 목표 변수는 연속적
즉, model의 output은 실제

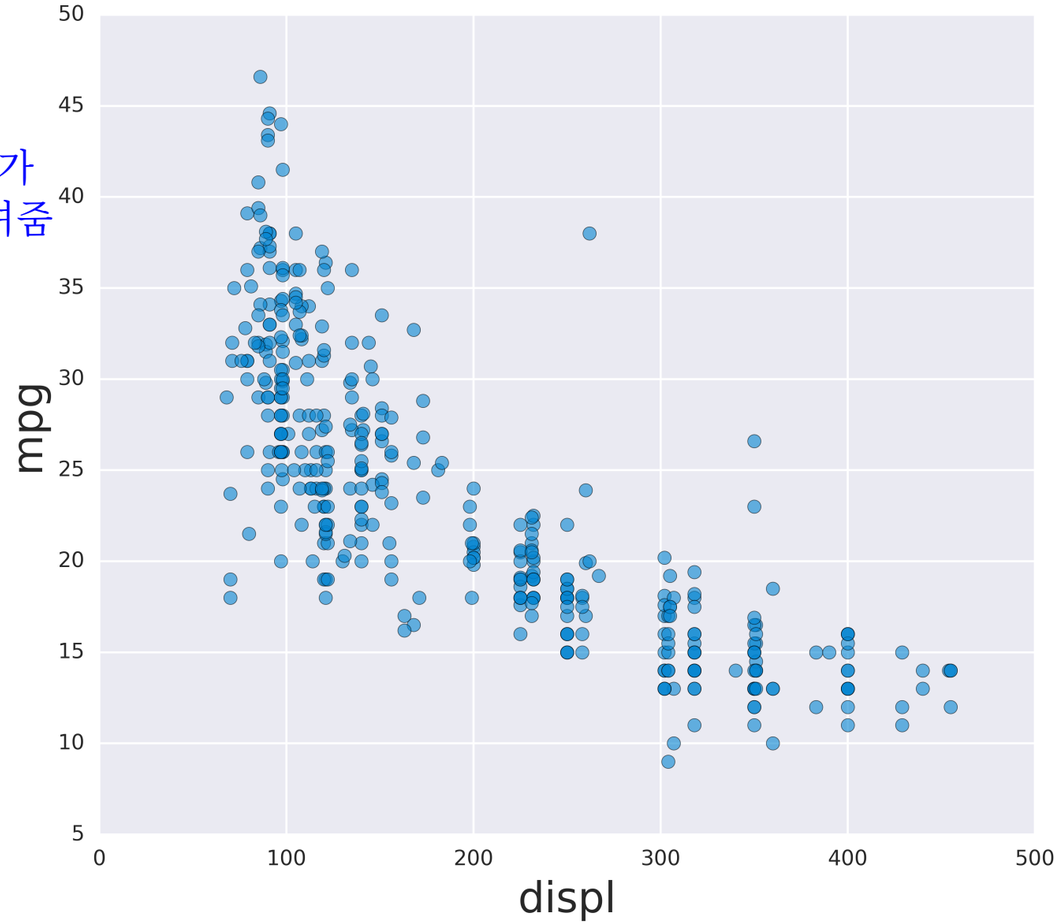
Auto-mpg Dataset

이 dataset는 자동차 특성에 해당하는 6가지 feature과 gallon 당 mile 수를 나타내는 mpg라는 연속 대산 변수로 구성
=> 해야할 것 : 6가지 feature를 고려해 자동차의 MPG 소비량을 예측

	mpg	displ	hp	weight	accel	origin	size
0	18.0	250.0	88	3139	14.5	US	15.0
1	9.0	304.0	193	4732	18.5	US	20.0
2	36.1	91.0	60	1800	16.4	Asia	10.0
3	18.5	250.0	98	3525	19.0	US	15.0
4	34.3	97.0	78	2188	15.8	Europe	10.0
5	32.9	119.0	100	2615	14.8	Asia	10.0

Auto-mpg with one feature

mpg대 displ의 2D scatter는 mpg 소비가 변위에 따라 비선형적으로 감소함을 보여줌



Regression-Tree in scikit-learn

```
# Import DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor
# Import train_test_split
from sklearn.model_selection import train_test_split
# Import mean_squared_error as MSE
from sklearn.metrics import mean_squared_error as MSE
# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test= train_test_split(X, y,
                                                   test_size=0.2,
                                                   random_state=3)

# Instantiate a DecisionTreeRegressor 'dt'
dt = DecisionTreeRegressor(max_depth=4,
                           min_samples_leaf=0.1,
                           random_state=3)
```

min_samples_leaf를 0.1로 설정하여 각 leaf가 훈련 데이터의 적어도 10% 이상을 포함해야하는 정지 조건을 설정

파라미터 명	설명
min_samples_split	- 노드를 분할하기 위한 최소한의 샘플 데이터수 → 과적합을 제어하는데 사용 - Default = 2 → 작게 설정할 수록 분할 노드가 많아져 과적합 가능성 증가
min_samples_leaf	- 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수 - min_samples_split과 함께 과적합 제어 용도 - 불균형 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 작게 설정 필요
max_features	- 최적의 분할을 위해 고려할 최대 feature 개수 - Default = None → 데이터 세트의 모든 피처를 사용 - int형으로 지정 →피처 갯수 / float형으로 지정 →비중 - sqrt 또는 auto : 전체 피처 중 √(피처개수) 만큼 선정 - log : 전체 피처 중 log2(전체 피처 개수) 만큼 선정
max_depth	- 트리의 최대 깊이 - default = None → 완벽하게 클래스 값이 결정될 때 까지 분할 또는 데이터 개수가 min_samples_split보다 작아질 때까지 분할 - 깊이가 깊어지면 과적합될 수 있으므로 적절히 제어 필요
max_leaf_nodes	리프노드의 최대 개수

Regression-Tree in scikit-learn

```
# Fit 'dt' to the training-set
dt.fit(X_train, y_train)
# Predict test-set labels
y_pred = dt.predict(X_test)
# Compute test-set MSE
mse_dt = MSE(y_test, y_pred)
# Compute test-set RMSE
rmse_dt = mse_dt**(1/2)
# Print rmse_dt
print(rmse_dt)
```

test_set에서 평균 제곱근 오차를 얻기 위해

먼저 평균 제곱 오차를 평가하고

그런 다음 얻은 값을 1/2의 거듭 제곱으로 올림

마지막으로 dt의 test set rmse를 인쇄하여 5.1의 값을 얻음

```
5.1023068889
```

Information Criterion for Regression-Tree

$$I(\text{node}) = \underbrace{\text{MSE}(\text{node})}_{mean-squared-error} = \frac{1}{N_{node}} \sum_{i \in node} \left(y^{(i)} - \hat{y}_{node}\right)^2$$

회귀 tree가 dataset에 대해 훈련될 때 node의 불순물은 해당 node에 있는 대상의 평균 제곱 오차를 사용하여 측정된다는 점을 유의
즉, 회귀 트리는 각 leaf의 목표값이 평균이고 특정 잎의 label 평균 값에 가장 가까운 잎을 생성하는 분할을 찾으려고 함

$$\underbrace{\hat{y}_{node}}_{mean-target-value} = \frac{1}{N_{node}} \sum_{i \in node} y^{(i)}$$

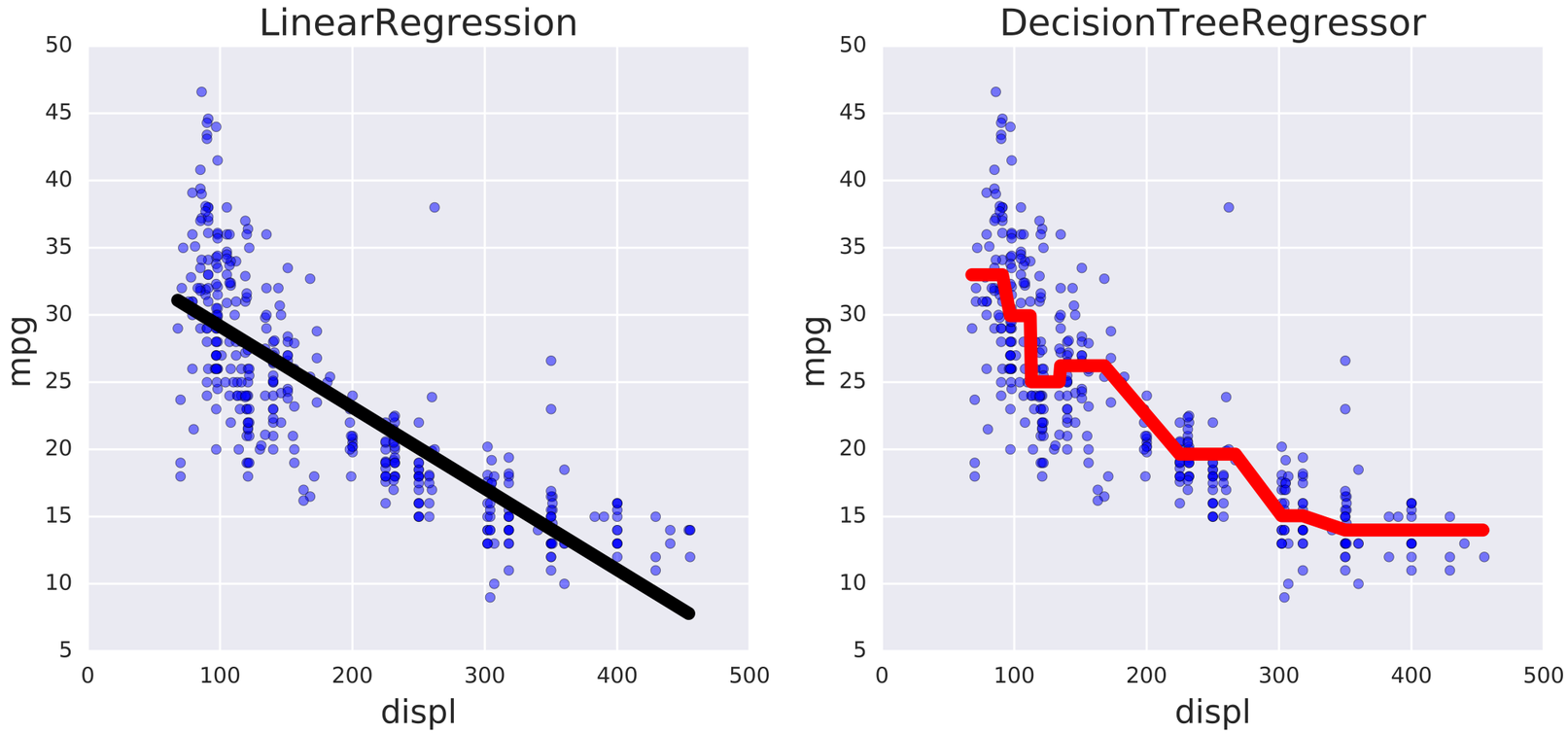
목표 값은 평균이며, 특정 잎에 있는 레이블의 평균 값에 가능한 가장 가까운 것임

Prediction

$$\hat{y}_{pred}(\text{leaf}) = \frac{1}{N_{leaf}} \sum_{i \in leaf} y^{(i)}$$

새 instance가 tree를 가로 질러 특정 leaf에 도달하면 대상 변수 'y'는
이 공식에 표시된대로 해당 leaf에 포함 된 대상 변수의 평균으로 계산

Linear Regression vs. Regression-Tree



회귀 tree의 유연성의 중요성을 강조하려면 이 그림을 살펴봐야함
왼쪽에는 파란색 data의 scatter이고 검은색으로 선형 회귀 모형의 예측이 표시되어 있음
(선형 모형은 데이터에 의해 나타나는 비선형 추세를 포착하지 못함)

오른쪽에는 앞서 훈련한 회귀 트리에 해당하는 빨간색선과 동일한 scatter
regression tree는 유연성이 뛰어나며 완전히는 아니지만 비선형성을 포착할 수 있음

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON