Decision Tree
결정 나무

Dr. Saerom Park
Statistical Learning and Computational Finance Lab.
Department of Industrial Engineering
psr6275@snu.ac.kr
http://slcf.snu.ac.kr

목차

Reference

- **Reading:** [Raschka. (2017), chapter 3], [Müller. (2016), chapter 2], [GÉRON. (2017), chapter 3 & 5 & 6]

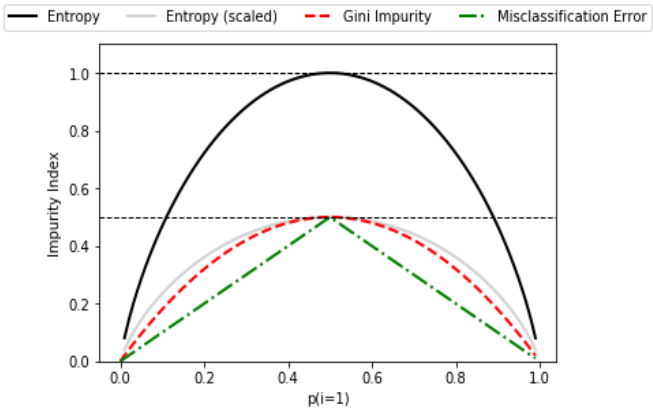
분류 결정 나무

의사결정 나무(Decision tree)

의사결정 나무

- 의사결정 나무는 데이터들을 한 개에 노드에 할당하고 특정 변수들을 기준으로 나누면서 각 노드의 데이터들이 충분히 homogeneous 지거나 나무가 최대 깊이에 도달했을 때 종료한다.
  - IG(Information gain = impurity의 감소량)을 최대화하는 방향으로 node를 만든다.
    - $IG(D_p, f) = I(D_p) - \sum_j \frac{N_j}{N_p} I(D_j)$

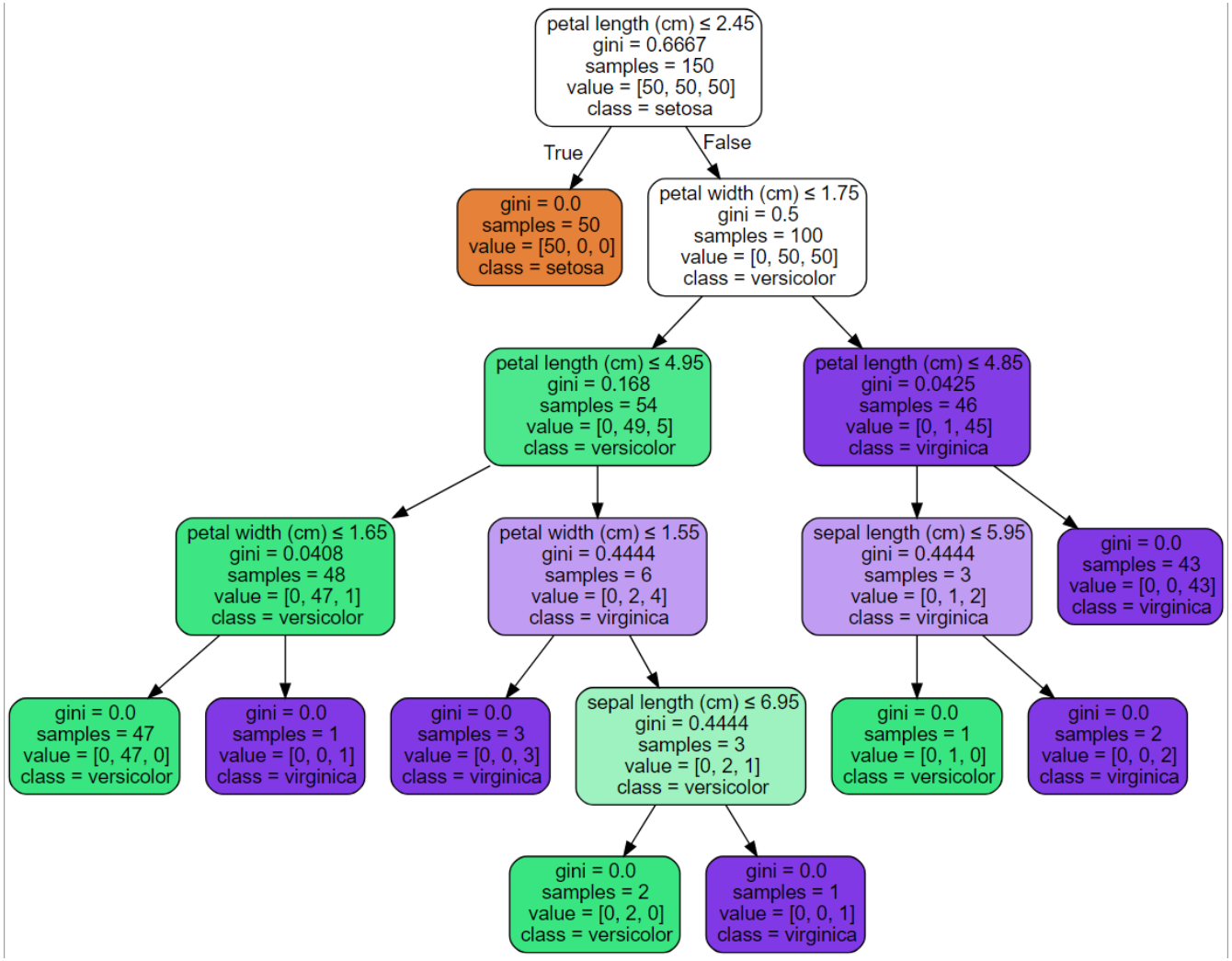
      $where\ D_p, D_j: parent, jth\ dataset; f: feature\ to\ perform\ the\ split$
  - Impurity
    - Gini impurity: $I_G(t) = \sum_i p(i|t)(1 - p(i|t))$
    - Misclassification error: $I_E(t) = 1 - \max_i\{p(i|t)\}$
    - Entropy: $I_H(t) = -\sum_i p(i|t) \log_2 p(i|t)$



- 장점:
  - 이해하고 해석하고 학습하기 쉬움
  - 모델이 유동적이고 강력함
- 단점:
  - 수직인 결정 경계만 학습 가능 (한번에 한 개의 변수 고려)
  - 데이터에 민감 (데이터 회전, 일반화)
  - Random forest로 극복

의사결정 나무(Decision tree)

- Example



의사결정 나무(Decision tree)

- scikit-learn 이용 의사결정 나무
  - `from sklearn.tree import DecisionTreeClassifier`
  - `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, min_impurity_split=None, class_weight=None, presort=False)`

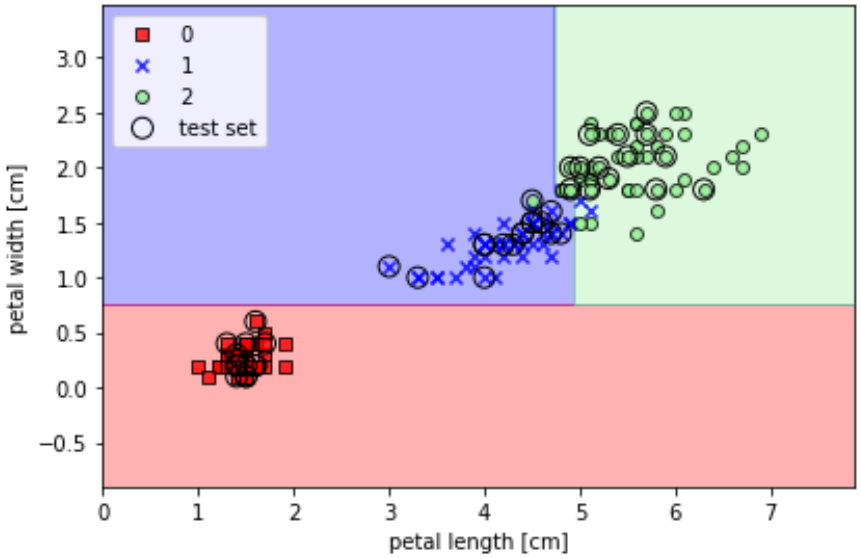
http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Parameters	
criterion	알고리즘에 사용하는 impurity('gini', 'entropy')
max_depth	Tree의 최대깊이
max_features	Split을 얻기 위해 사용하는 features 수

Attributes	
classes_	분류 라벨 array
tree_	최종 tree

의사결정 나무(Decision tree)

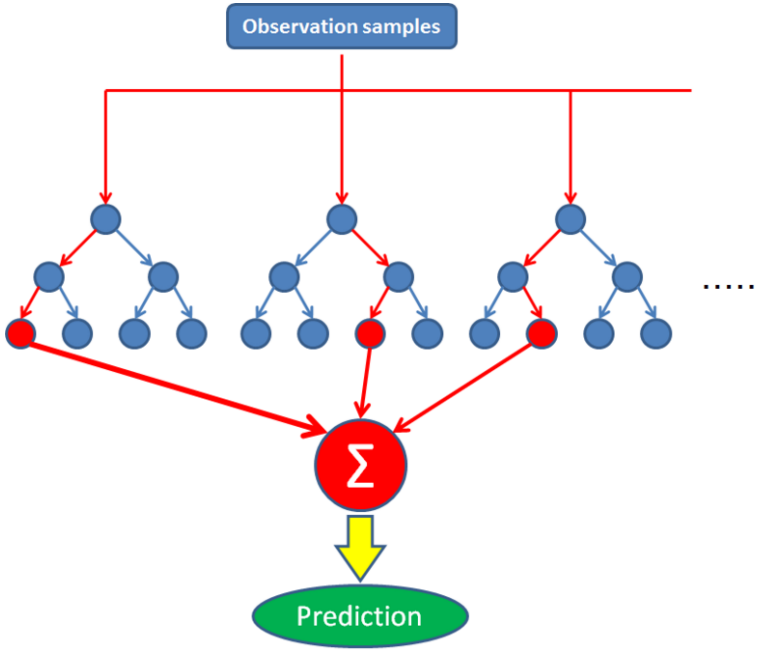
- scikit-learn 이용 의사결정 나무
  - `from sklearn.tree import DecisionTreeClassifier`
  - `DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)`



앙상블 방법

랜덤 포레스트(Random Forest)

- 랜덤 포레스트
  - Bootstrapping: n개의 samples을 복원추출
  - $d = \sqrt{m}$ features (m : 총 features 수)를 임의로 선택하여 DT와 같은 방법으로 가지를 나눔
    - 반복(또는 병렬) 학습 가능
  - 다수결(또는 평균)



  - 여러 DT를 이용해서 정확도를 높이고 오버피팅 문제를 해결할 수 있다.
  - 변수 선택에도 활용

랜덤 포레스트(Random Forest)

- scikit-learn 이용 랜덤 포레스트
  - `from sklearn.ensemble import RandomForestClassifier`
  - `RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)`

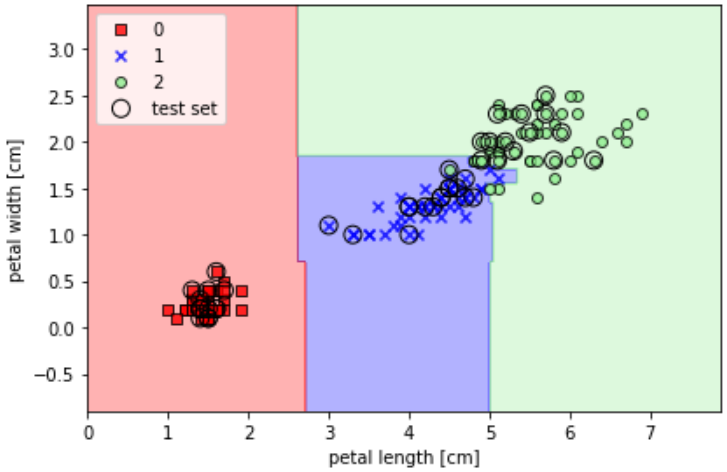
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Parameters	
n_estimators	forest를 이루는 tree 수
criterion	알고리즘에 사용하는 impurity('gini', 'entropy')
max_depth	Tree의 최대깊이
max_features	Split을 얻기 위해 사용하는 features 수
bootstrap	bootstrap 여부
n_jobs	병렬처리 방법

랜덤 포레스트(Random Forest)

- scikit-learn 이용 랜덤 포레스트
  - `from sklearn.ensemble import RandomForestClassifier`
  - `RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)`

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html



Random Forest: Feature Selection

- 모델 기반 특성 선택
  - Random forest란 여러 개의 decision tree들을 임의적으로 학습하는 앙상블 방법(ensemble method)이다.
  - Random forest를 이용해 분류 또는 회귀 문제에서 각 feature의 중요성에 순위를 매길 수 있다.
  - Scikit-learn에서는 random forest classifier, regressor를 제공할 뿐만 아니라 feature_importances_ 어트리뷰트를 통해서 feature 중요성 값을 제공한다.

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) [source]

feature_importances_ : array of shape = [n_features]

The feature importances (the higher, the more important the feature).

- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

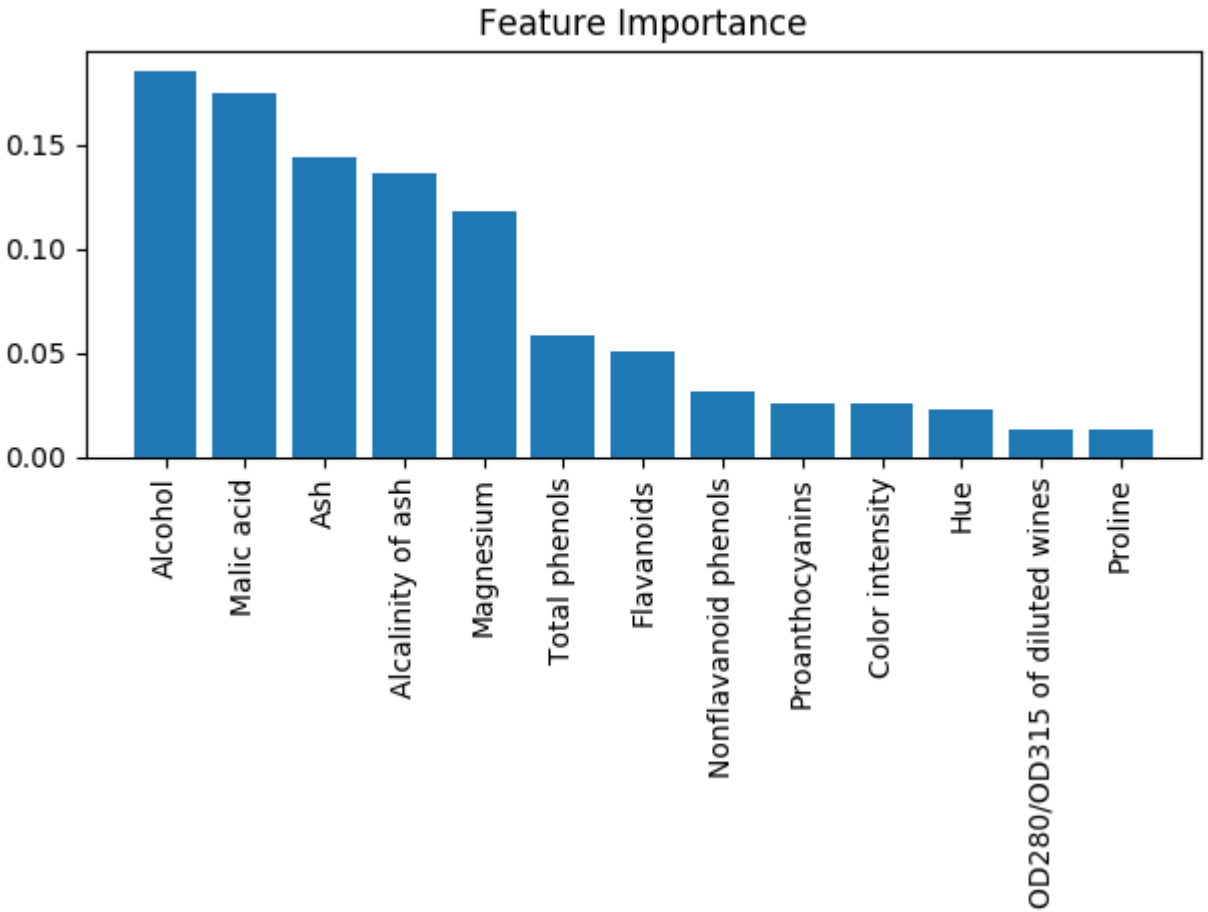
Random Forest: Feature Selection

- Assessing feature importance with random forests
  - 실행 결과

```
 1) Proline                        0.185453
 2) Flavanoids                     0.174751
 3) Color intensity                0.143920
 4) OD280/OD315 of diluted wines   0.136162
 5) Alcohol                        0.118529
 6) Hue                            0.058739
 7) Total phenols                  0.050872
 8) Magnesium                      0.031357
 9) Malic acid                     0.025648
10) Proanthocyanins                0.025570
11) Alcalinity of ash              0.022366
12) Nonflavanoid phenols           0.013354
13) Ash                            0.013279
```

Random Forest: Feature Selection

- Assessing feature importance with random forests
  - 실행 결과



Random Forest: Feature Selection

- SelectFromModel
  - sklearn.feature_selection.SelectFromModel
  - 중요도가 지정한 임계치보다 큰 모든 특성을 선택한다.
  - http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html

Boosting

- Hypothesis boosting
  - Ensemble이 성능이 낮은 학습기(weak learners)들로 구성되어 있을 때, boosting 을 통해 ensemble의 성능을 향상시킨다.
  - 모델들을 순차적으로 학습시킴
  - Boosting은 bagging에 비해 낮은 bias와 variance를 만들어낼 수 있다.
- AdaBoost(Adaptive Boosting)
  - 분류가 힘든 트레이닝 샘플에 집중하여 weak learner들이 잘못 분류된 샘플로부터 학습하도록 한다.
  - 학습된 weak learner들의 예측 결과들을 aggregation함
- Gradient Boosting
  - 새로운 예측기가 이전의 예측기의 residual errors을 학습 하도록 함
  - 회귀 문제에 적합
  - 순차적으로 학습된 예측기들의 예측 결과들을 합함으로써 최종 결과를 예측

Adaptive Boosting

Adaboost 분류기

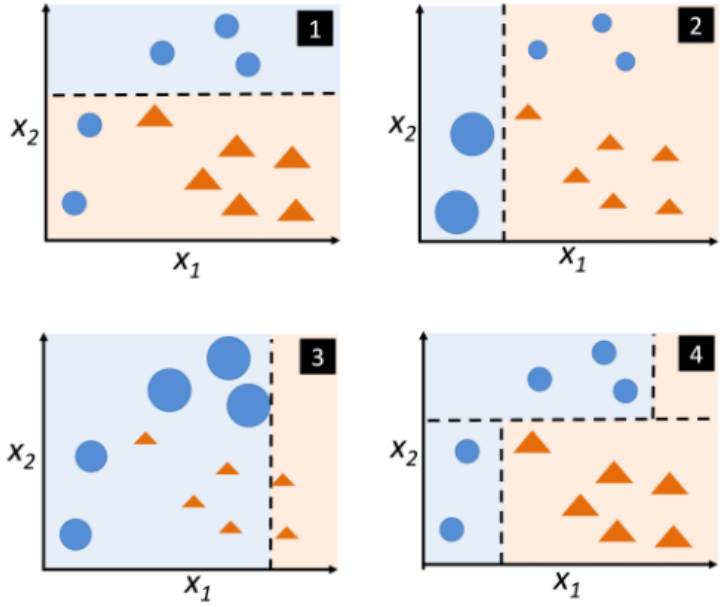
- 트레이닝 셋 D가 주어짐 (sampling without replacement):
  - random subset d1을 추출하여 weak learner C1을 학습
  - random subset d2와 이전에 잘못 분류된 샘플 중 50%를 더하여 weak learner C2를 학습
  - C1, C2가 서로 다른 결과를 반환한 데이터들로 d3를 구성하여 weak learner C3를 학습
- Majority voting을 통해 C1, C2, C3를 ensemble한다.

Adaboost 알고리즘

- Weight 벡터 w를 모두 같게 설정한다. ($\sum_i w_i = 1$)
- 총 m번의 boosting rounds 중 j번째에서, 다음을 실행한다.
  - Weighted된 weak learner $C_j = train(X, y, w)$를 학습시킨다.
  - 클래스 label을 예측한다 : $\hat{y} = predict(C_j, X)$
  - Weighted 된 에러 비율을 계산한다 : $\epsilon = w \cdot \mathbb{1}_{\hat{y} \neq y}$
  - 계수를 계산한다 : $\alpha_j = 0.5 \log \frac{1-\epsilon}{\epsilon}$
  - Weight들을 업데이트한다 : $w := w \times \exp(-\alpha_j \times \hat{y} \times y)$
  - 합이 1이 되도록 weight 를 표준화한다 : $w := w / \sum_i w_i$

Adaptive Boosting

- Adaboost 분류기 학습 예제
  - 이 때, 앞 단계의 오분류로부터 학습하여 성능을 높이기 위해 각 샘플들의 weight가 단계마다 달라짐
    - 그림 1에서 모든 샘플의 weight가 같으므로 점선과 같이 학습
    - 그림 2에서 오분류된 두 샘플의 weight를 높여서 학습
    - 그림 3에서 새롭게 오분류된 세 샘플의 weight를 높이고, 앞에서 오분류되었던 샘플의 weight를 낮추어 학습
  - 3개의 분류기를 ensemble하면 그림 4와 같이 정확한 분류기 학습이 가능



Adaptive Boosting

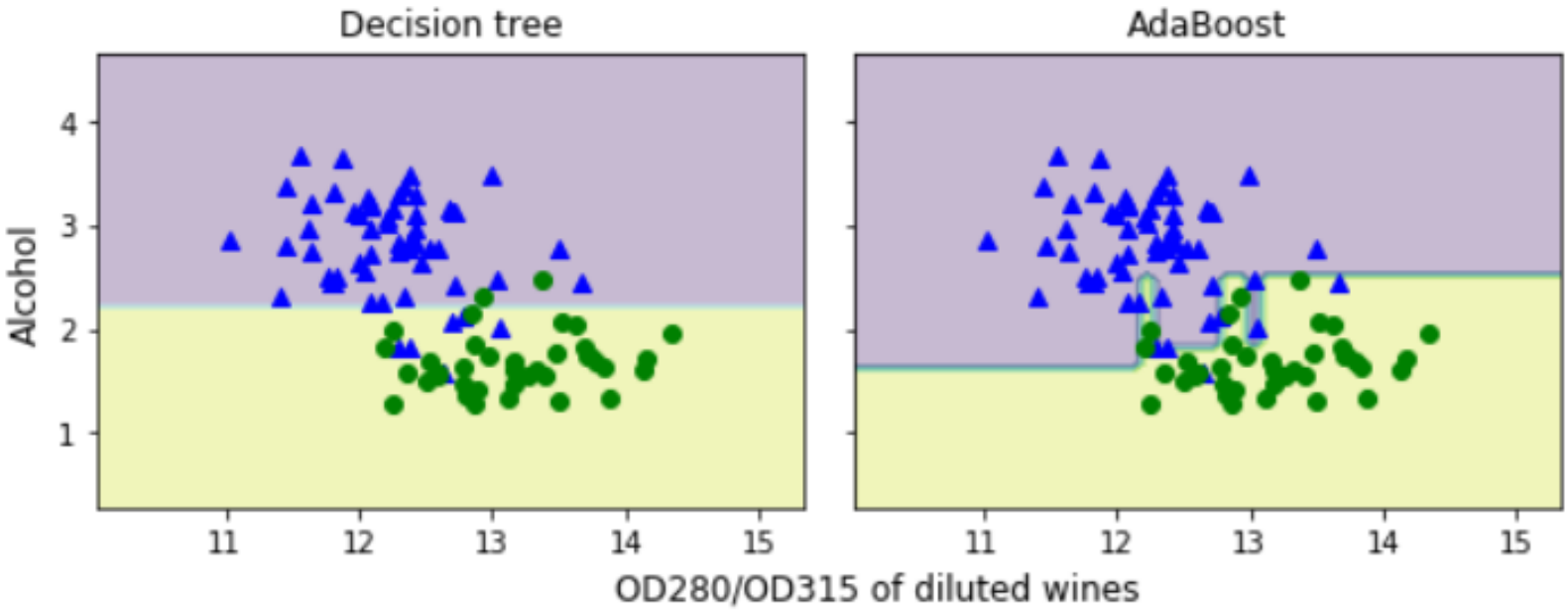
- Sklearn에서 Adaboost 사용 방법
  - sklearn.ensemble.AdaBoostClassifier

Parameters	
base_estimator	The base estimator from which the boosted ensemble is built
n_estimators	The maximum number of estimators at which boosting is terminated
learning_rate	Learning rate shrinks the contribution of each classifier

  - 자세한 사용법은 http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html 참조

Adaptive Boosting

- 예제 코드



- AdaBoost 모델의 decision boundary가 더 복잡하며, 앞에서의 bagging과 유사한 형태이다.

회귀 결정 나무

Non-Linear regression

Decision tree regression

- Decision tree regression은 비선형 데이터를 다루기 위해 앞과 같은 어떤 transformation도 필요 없다. 앞서 나온 바와 같이 Node와 Leave를 통해 분류하기 때문이다.
- 이 때 Information Gain(IG)를 정의하여, IG를 최대화하도록 Decision tree를 구현한다.

$$IG(D_p, x_i) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

x : spilt를 진행할 변수
I : impurity 함수
N_p : parent node의 sample 개수
D_P : parent node의 training sample의 부분집합
D_{left} : left child node의 training sample의 부분집합
D_{right} : right child node의 training sample의 부분집합

Non-Linear regression

Decision tree regression(Cont')

- 회귀 문제에서는 $I(t)$으로 MSE를 사용하기도 한다.

$$I(t) = MSE(t) = \frac{1}{N_t} \sum_{i \in D_t} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$$\hat{y}^{(i)} = \frac{1}{N_t} \sum_{i \in D_t} y^{(i)}$$

D_t : node t의 training sample 부분집합

Random forest regression

- 앞서 배운 Forest Regression과 Random forest 알고리즘을 활용하여, Random forest regression을 실행할 수 있다.

Non-Linear regression

- Scikit-learn의 Random forest regression
  - sklearn.ensemble.RandomForestRegressor(n_estimators=10, criterion='mse', max_depth=None, … , warm_start=False)

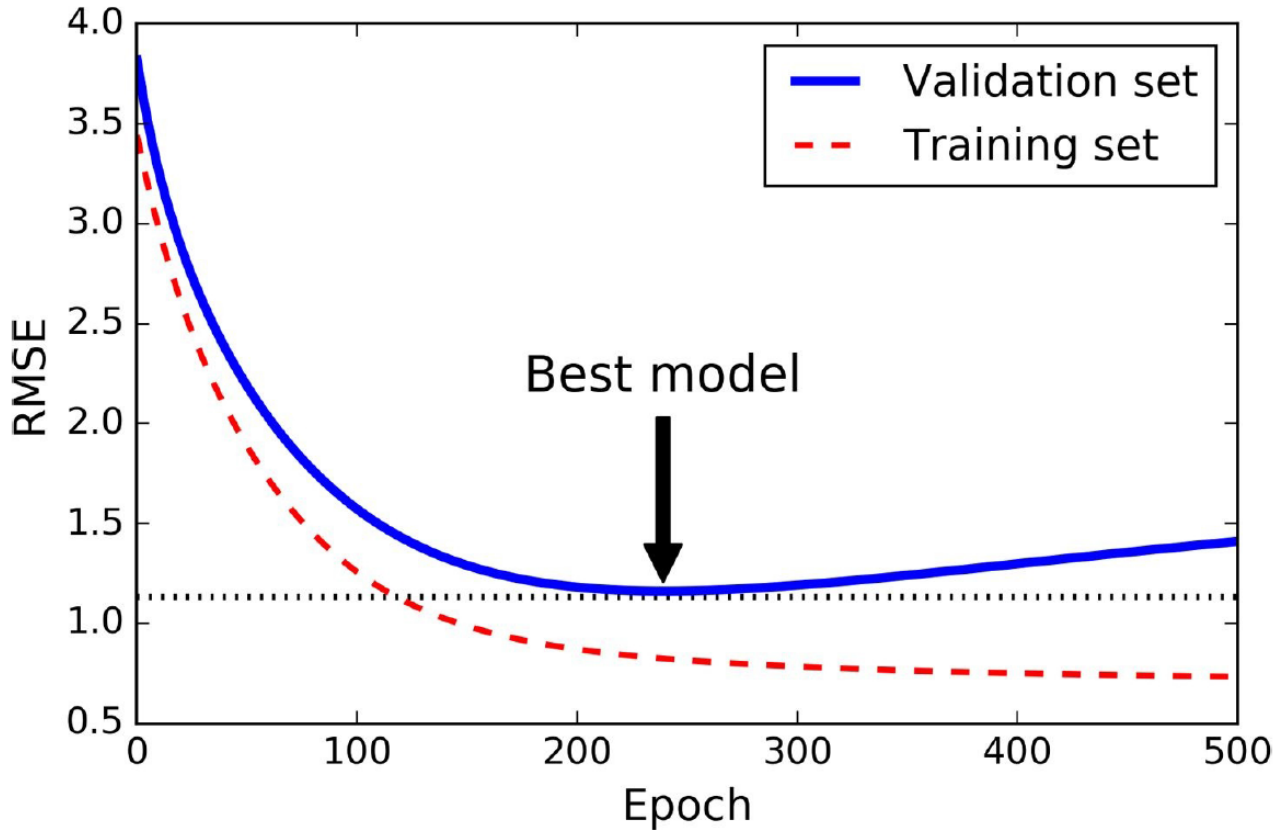
Parameters	
n_estimators	포레스트에 사용될 트리의 개수
criterion	분류의 질을 계산할 방법. Mse는 mean squared error. Mae는 mean absolute error
max_depth	트리의 최대 깊이
warm_start	여러 CPU를 병행하여 사용할 것인가에 대한 여부 (-1이면 모두 사용)

Attributes	
estimators_	DecisionTreeRegressor의 리스트를 반환
n_features_	사용된 특성의 개수를 반환

  - http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Early Stopping

- Early Stopping



- 경우에 따라 학습을 오래하면 할수록 오히려 MSE가 증가할 수도 있다. 이 때에는 증가하기 전, 최소점이 Best Model이므로 Early Stopping이 요구된다.

Practice

- Non-linear Regression
  - Turning a linear regression model into a curve - polynomial regression
  - Modeling nonlinear relationships in the Housing Dataset
  - Decision tree regression
  - Random forest regression
  - Code 03 참고

Reference

- [Raschka. (2017)] Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning*. Packt Publishing Ltd, 2017.
- [Müller. (2016)] Müller, Andreas C., and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. 2016.
- [GÉRON. (2017)] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Sebastopol, CA: O, 2017.