주제 : Ensemble Learning

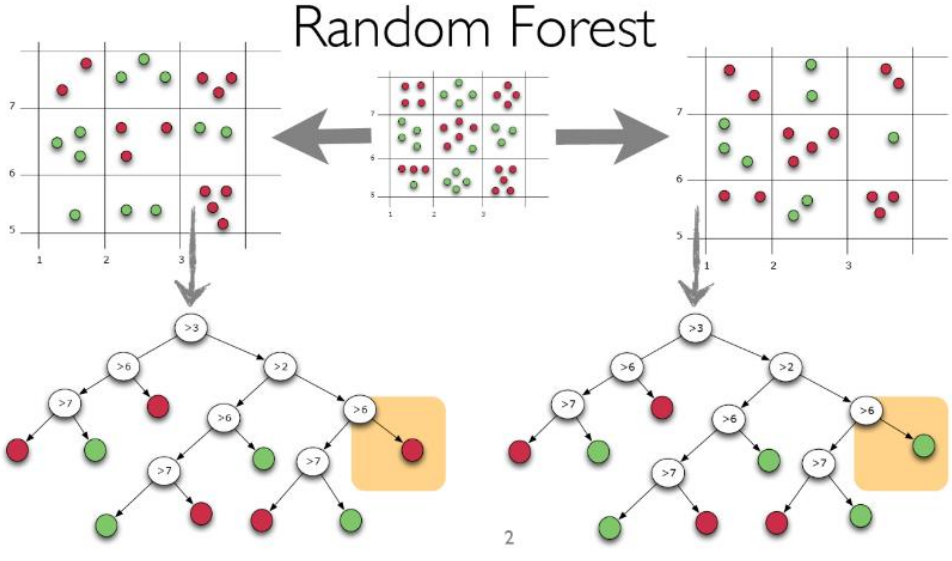
(논문 : Variable Importance Assessment in Regression: Linear Regression versus Random Forest)

해당 포스팅은 고려대 강필성 교수님의 강의 내용 중에서 OOB(Out of Bag)을 사용하여 Random forest 알고리즘에서 Variable Importance 확인 방법이 신선하고 재미있어 보여서 시작하게 되었다. 다양한 내용이 존재하지만 해당 내용에서는 Random Forest 알고리즘과 OOB를 통한 Variable Importance 확인에 집중해서 작성한다.(고려대 강필성 교수님의 자료 내용이 포함되어 있음을 밝힙니다.)

1. Random Forest 란?

Decision Tree는 Train Data에 대해 Overfitting 가능성이 높다는 단점을 가지고 있다. Pruning등의 방법이나 가지 생성에 제한을 가하여 어느정도 Overfitting을 막을 수는 있으나, 여전히 충분하지 못하다. 그래서 좀더 일반화된 Tree를 만들려고 시도하는 모델이 바로 Random Forest의 아이디어다.

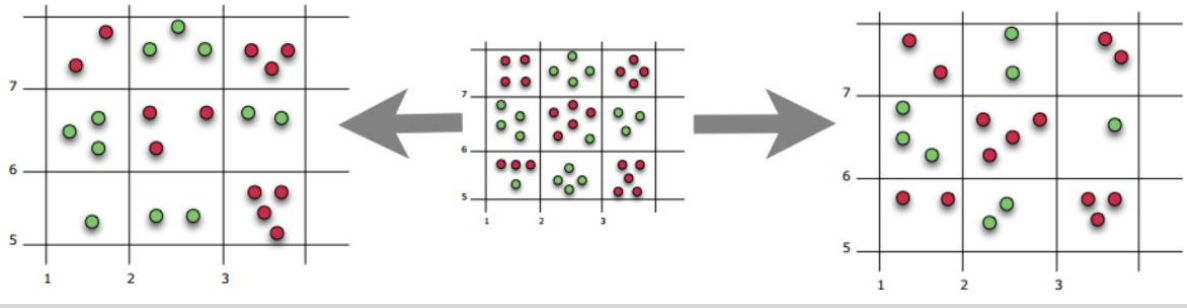
Random Forest는 Ensemble(앙상블) Learning 모델이다. 단일 Tree만을 구성하는 Decision Tree와는 다르게 여러 개의 Decision Tree로 구성된다. 그리고 새로운 데이터 포인트를 각 Tree에 동시에 적용하여, 각 Tree가 분류한 결과에서 Majority Voting 을 통해 최종 결과를 결정하게 된다.



위의 그림은 고려대 강필성 교수님의 강의 자료에서 가져온 것이며, Random Forest 알고리즘의 핵심은 알고리즘 명에서 잘 나타내고 있다. Decision Tree 알고리즘 기반의 bagging의 특별한 형태이면서 bagging 방식과 예측 변수를 Randomly 선택한다는 것이 Random Forest 알고리즘의 핵심이다.

2. Bagging ?

Random Forest는 Bagging이라는 과정을 수행한다. Bagging은 Tree를 구성함에 있어서 Training Data Set의 부분집합을 활용하는 것을 말한다. 아래 가운데 그림을 보게 되면 Training Data Set을 임의로 X개의 데이터만으로 각 Tree를 구성할 때 활용한다. 즉, 모든 Tree는 각기 다른 데이터를 바탕으로 형성되지만 모두 Training Data Set의 부분집합이라는 말이다. 또한 Training Date Set에서 임의 선택에 있어서 중복을 허용함으로써 각 Tree를 형성한다.



Random Forest는 Tree 형성시에 Training Data Set 뿐만 아니라 Feature 선택을 통해 변화를 가지고 온다. Feature 선택은 Training Data Set 선택과 마찬가지고 Feature들의 부분집합을 활용하게 된다.

3. Python Code 구현

크게 3개의 부분으로 구성했다.

- **make_random_Forest**
- **make_tree**
- **find_best_split_subset**

◦ **make_random_Forest**

아래 코드는 설정하는 n개의 tree(n_tree)에 대해 Training Data Set을 make_tree 함수로 전달하는 코드이다. 해당 진행에서는 feature는 2로 진행하였다.

```
def make_random_forest(n_tree, X_train, Y_train):
    # 트리 저장
    trees = []

    # n만큼 트리 형성
    for i in range(n_tree):
        indices = [random.randint(0, len(X_train)-1) for x in range(len(X_train))]

        X_train_subset = [X_train[index] for index in indices]
        Y_train_subset = [Y_train[index] for index in indices]

        tree = make_tree(X_train_subset, Y_train_subset, 2)
        trees.append(tree)

    return trees
```

◦ **make_tree**

information 기반으로 최적의 Subset을 찾고, 해당 feature(best_feature)와 gain(best_gain)을 기반으로 종료 조건과 최적 feature에 대한 각 하위 Decision Tree 를 생성을 수행한다.

```
def make_tree(X, Y, n_feature, value=""):
    # information_gain을 기반으로 최적의 subset을 찾음
    best_feature, best_gain = find_best_split_subset(X, Y, n_feature)

    # node 분할의 종료 조건으로 information_gain이 0에 근사하면 종료한다.
    if best_gain < 0.0001:
        return Leaf(Counter(Y), value)

    # best feature로 분할 수행
    X_subset, Y_subset = split(X, Y, best_feature)
    branches = []

    # 분할 후 subset에 대해 make_tree 함수 재귀 호출 수행
    for i in range(len(X_subset)):
        branch = make_tree(X_subset[i], Y_subset[i], n_feature, X_subset[i][0][best_feature])
        branches.append(branch)

    return Internal_Node(best_feature, branches, value)
```

◦ **find_best_split_subset**

입력 데이터의 주어지는 총 10개의 feature에서 3개를 중복허용으로 임의로 선택하여 해당 feature에 대해 최적의 information_gain과 feature를 확보한다.

```python
def find_best_split_subset(best_X, best_Y, best_n_features):
    # 총 10개의 feature에서 3개의 feature를 중복 허용으로 임의로 선택
    features = np.random.choice(10, 3, replace=True)
    best_gain = 0
    best_feature = 0

    for feature in features:
        X_subset, Y_subset = split(best_X, best_Y, feature)
        gain = information_gain(best_Y, Y_subset)

    if gain > best_gain:
        best_gain, best_feature = gain, feature

    return best_feature, best_gain
```

4. Variable Importance

원본 OOB Data와 i번째 변수에 대해 Random Permutation이 수행된 OOB Data 두가지 Data를 통해 확보한 Error 값을 통해 Error의 변화가 큰 i번째 변수는 해당 Random Forest 모델 생성에 있어서 Variable Importance라고 판단하는 방법이다.

Random Forest에서 Variable Importance가 높다면 Random Permutation 전-후의 OBB Error 차이가 크게 나타나고, 그 차이의 편차가 작아야 한다. 수식으로 표현하면 아래와 같다.

$$d_i{}^m = p_i{}^m - e_i{}^m$$

위 수식은 m번째 Tree에서 변수 i에 대한 Random Permutation 전-후 OOB Error의 차이를 나타낸다.

$$\bar{d}_i = \frac{1}{m}\sum_{i=1}^{m} d_i{}^m \qquad s_i{}^2 = \frac{1}{m-1}\sum_{i=1}^{m}\left(d_i{}^m - \bar{d}_i\right)^2$$

위 수식은 전체 Tree 들에 대한 OOB Tree Error 차이의 평균 및 분산에 대해 나타낸 값이다. 해당 값들을 통해 아래 수식처럼 Variable Importance를 추정한다.

$$v_i = \frac{\bar{d}_i}{s_i}$$

위 값 v_i는 i번째 변수 중요도를 나타내고, 해당 값은 상대적인 값으로 절대값이 가지는 크기의 의미는 없다.

Variable Importance를 sklearn 라이브러리를 통해 구현한 코드이다. iris 데이터를 통해 확인하였고, 해당 데이터의 Variable Importance 의 결과도 같이 출력해 보았다.

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier

iris = load_iris()
model = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=100)
model.fit(iris['data'], iris['target'])
model.feature_importances_
```

```
model.feature_importances_
array([0.09294531, 0.01751136, 0.44775647, 0.44178686])
```

5. 느낀 점

Ensemble Learning에 대해 배우면서 현업을 함에 있어서도 맥락이 일맥상통한다고 생각한다.

Decision Tree 단일 모델로는 문제나 한계점이 발생하기에 이를 극복하기 위해서 Ensemble 개념을 도입해본 것처럼 개개인의 각자의 능력도 중요하지만 어쩌면 한명이서 모든 일을 전부 처리할 수 없는 것처럼 집단지성을 통해 해결하는 방법이었다. 그리고 조직에 있어서도 다양한 분야의 전문가들이 모여서 일을 처리하게 되었을 때의 강점을 연결 시켜본다.

그렇게 집단 지성과 다양성을 인정하기 시작하면, Random Forest에서 Variable Importance처럼 부가적인 산출물들을 많이 만들 수 있을 것이라 기대해본다.