2020 D&A

MachineLearning SESSION

Ensemble(Bagging)

D&A

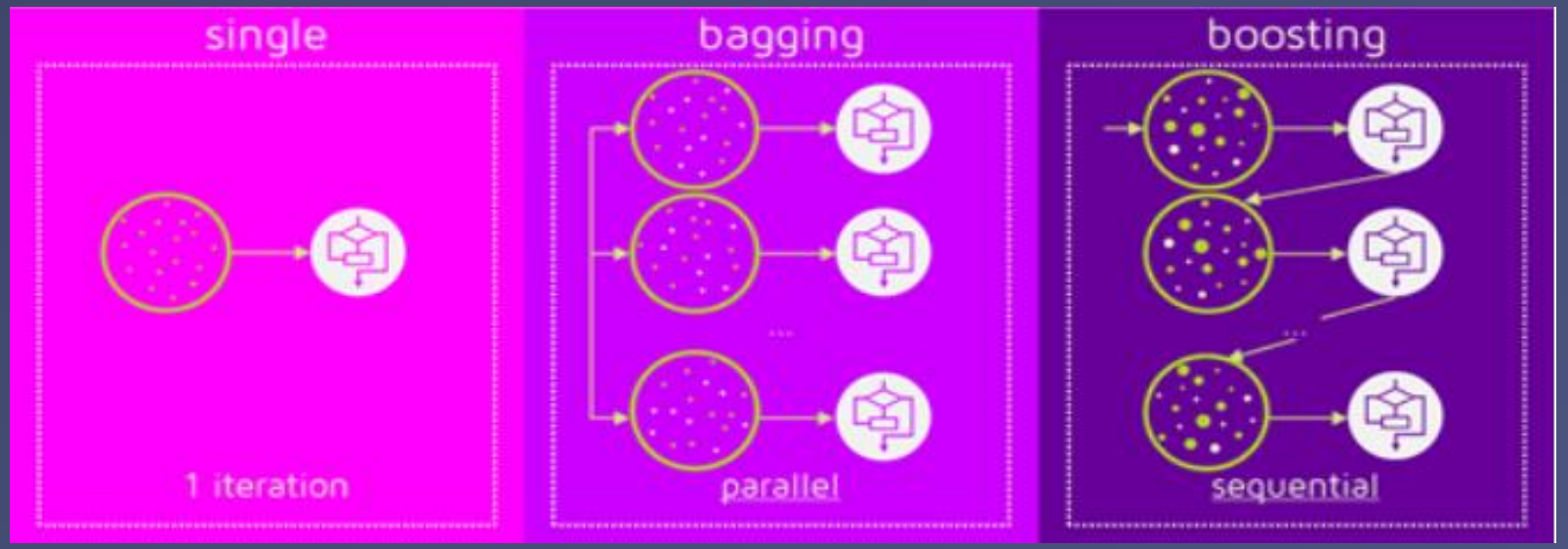
Contents

1. Ensemble Method
2. Bagging
3. RandomForest

Ensemble Method

1. Bagging : 일반적인 모형을 만드는데 초점, 분산을 줄여 과대적합(Overfitting)을 막아줌

2. Boosting : 맞추기 어려운 문제를 맞추는 데 초점, 틀린 문제에 가중치 부과



D&A

Ensemble Method

3. Voting

1) Hard Voting : 예측한 결과값들 중 다수의 분류기가 결정한 예측값을

최종 결과값으로 선정(다수결)

2) Soft Voting : 분류기들의 예측값 결정 확률을 평균내어 가장 높은 레이블 값을

최종 결과값으로 선정

EX)

	20대	30대	40대
분류기1	10%	50%	40%
분류기2	10%	0%	90%
분류기3	30%	40%	30%

Ensemble Method

4. Stacking : 서로 다른 모델의 예측 결과값을 다시 학습 데이터로 만들어서 다른 모델로 재학습시켜 결과를 예측하는 방법

EX)

	20대	30대	40대
분류기1	10%	50%	40%
분류기2	10%	0%	90%
분류기3	30%	40%	30%

새로운 Feature!!

10	50	40	10	0	90	30	40	30

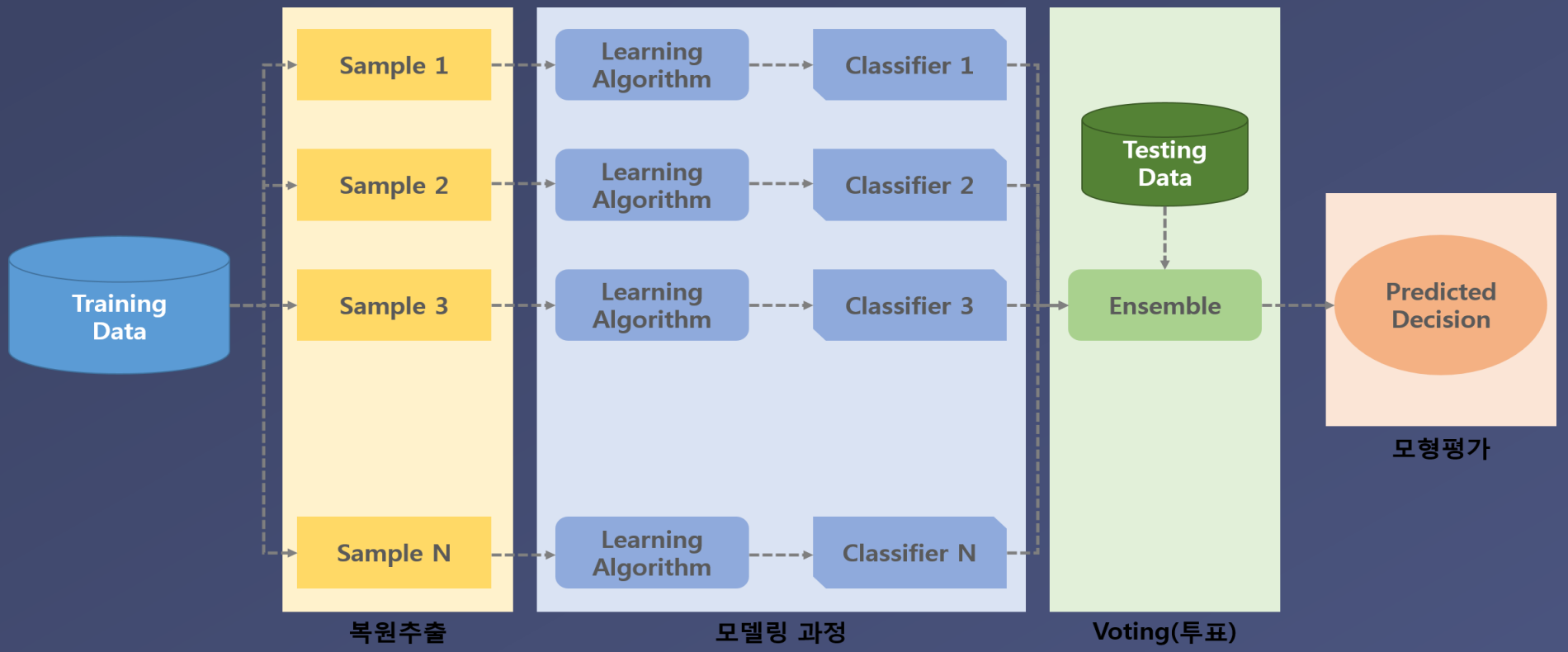
Ensemble Method

Why Ensemble?

1. 단일 모델로는 모델의 성능과 변동의 폭이 크다.
2. 모델의 분산을 줄여 일반화가 용이해짐.

Bagging

Bagging = Bootstrap + Aggregating



Bagging

1. Bootstrap sampling : 주어진 트레이닝 데이터셋에서 무작위로

중복을 허용하여 n번 샘플링

(Scikit-Learn이 제공하는 랜덤포레스트 API는 n=len(train))

	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	...	C70	C71	C72	C73	C74	C75	C76	C77	C78	C79
0	395.19528	12	10	52.80456	-1.2648	-1.87531	779.59595	28.02645	10832.0	-3.0660	...	808.29620	0.0	1.36810	8.79882	35.43700	12.01782	305.03113	301.35992	33.6555	6.0951
1	395.14420	12	10	52.78931	-1.3147	-1.88294	780.67328	28.02473	10984.0	-2.9721	...	819.16809	0.0	1.36810	8.78811	35.45227	12.01782	304.27161	297.43567	33.6555	5.9262
2	395.14420	12	10	52.79694	-1.4032	-1.88294	780.06574	28.02817	11120.0	-2.9857	...	823.51697	0.0	1.36734	8.81787	35.45227	12.01782	303.89179	298.66534	33.6555	5.8101
3	395.19528	12	10	52.79694	-1.6074	-1.88294	780.15265	28.02301	11256.0	-3.2166	...	823.95172	0.0	1.36734	8.87493	35.43700	12.01782	303.67474	298.06860	33.6555	5.7509
4	395.34866	12	10	52.79694	-1.7811	-1.88294	781.83160	28.03595	11384.0	-3.5613	...	827.86560	0.0	1.36810	8.83838	35.45227	12.01782	303.22266	296.53137	33.6555	5.8547
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
215996	380.26914	12	10	72.01538	-1.0303	97.18475	775.02948	28.02470	1144.0	-2.9797	...	927.01715	100.0	1.36353	15.09527	36.35254	12.01782	326.29846	334.90668	32.0000	5.1486
215997	380.32028	12	10	72.01538	-1.0989	97.18475	775.88190	28.02993	1776.0	-3.1034	...	933.54034	100.0	1.36429	15.13971	36.35254	12.02545	327.20270	335.08752	32.0000	5.1892
215998	380.37140	12	10	72.01538	-1.1531	97.19238	778.48596	28.02993	2384.0	-3.2595	...	929.62647	100.0	1.36276	15.15135	36.36779	12.02545	328.03461	336.67895	32.0000	5.2595
215999	380.47360	12	10	72.00774	-1.2477	97.19238	778.11762	28.03078	2960.0	-3.5434	...	926.14740	100.0	1.36353	15.17572	36.35254	12.02545	328.39624	335.55768	32.0000	5.4384
216000	380.52475	12	10	72.00774	-1.2521	97.19238	778.31903	28.02210	3504.0	-3.6989	...	937.45422	100.0	1.36353	15.16666	36.36779	12.01782	329.15582	336.24493	32.0000	5.5915

216001 rows × 79 columns

하나의 샘플씩 216001번

복원추출!!!

Bagging

2. Feature Selection : 1에서 선택한 데이터 샘플에서 중복 허용 없이

d개 만큼의 Feature 선택

(일반적으로 트레이닝 데이터의 전체 특성 개수를 m이라고

하면 d = m**(1/2))

	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	...	C70	C71	C72	C73	C74	C75	C76	C77	C78	C79
0	395.19528	12	10	52.80456	-1.2648	-1.87531	779.59595	28.02645	10832.0	-3.0660	...	808.29620	0.0	1.36810	8.79882	35.43700	12.01782	305.03113	301.35992	33.6555	6.0951
1	395.14420	12	10	52.78931	-1.3147	-1.88294	780.67328	28.02473	10984.0	-2.9721	...	819.16809	0.0	1.36810	8.78811	35.45227	12.01782	304.27161	297.43567	33.6555	5.9262
2	395.14420	12	10	52.79694	-1.4032	-1.88294	780.06574	28.02817	11120.0	-2.9857	...	823.51697	0.0	1.36734	8.81787	35.45227	12.01782	303.89179	298.66534	33.6555	5.8101
3	395.19528	12	10	52.79694	-1.6074	-1.88294	780.15265	28.02301	11256.0	-3.2166	...	823.95172	0.0	1.36734	8.87493	35.43700	12.01782	303.67474	298.06860	33.6555	5.7509
4	395.34866	12	10	52.79694	-1.7811	-1.88294	781.83160	28.03595	11384.0	-3.5613	...	827.86560	0.0	1.36810	8.83838	35.45227	12.01782	303.22266	296.53137	33.6555	5.8547
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
215996	380.26914	12	10	72.01538	-1.0303	97.18475	775.02948	28.02470	1144.0	-2.9797	...	927.01715	100.0	1.36353	15.09527	36.35254	12.01782	326.29846	334.90668	32.0000	5.1486
215997	380.32028	12	10	72.01538	-1.0989	97.18475	775.88190	28.02993	1776.0	-3.1034	...	933.54034	100.0	1.36429	15.13971	36.35254	12.02545	327.20270	335.08752	32.0000	5.1892
215998	380.37140	12	10	72.01538	-1.1531	97.19238	778.48596	28.02993	2384.0	-3.2595	...	929.62647	100.0	1.36276	15.15135	36.36779	12.02545	328.03461	336.67895	32.0000	5.2595
215999	380.47360	12	10	72.00774	-1.2477	97.19238	778.11762	28.03078	2960.0	-3.5434	...	926.14740	100.0	1.36353	15.17572	36.35254	12.02545	328.39624	335.55768	32.0000	5.4384
216000	380.52475	12	10	72.00774	-1.2521	97.19238	778.31903	28.02210	3504.0	-3.6989	...	937.45422	100.0	1.36353	15.16666	36.36779	12.01782	329.15582	336.24493	32.0000	5.5915

216001 rows × 79 columns

d = $\sqrt{79}$개의 Feature를

중복 없이 선택!!!

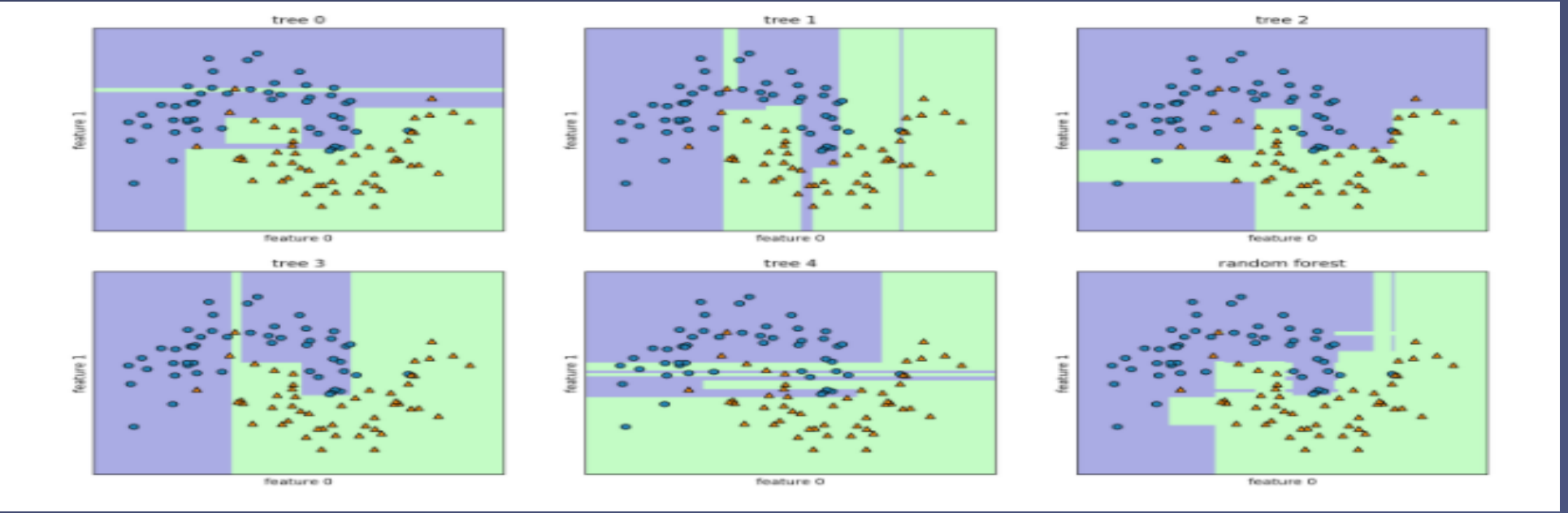
D&A

Bagging

3. 1과 2 과정에서 만들어진 데이터 샘플을 모델이 학습

4. 1~3단계를 k번 반복

5. 1~4단계를 통해 생성된 k개의 모델들의 결과를 집계(Aggregating)



RandomForest

RandomForest : Weak Classifier인 Decision Tree를 여러 개 결합한 앙상블 모델!!

1) 개별 Tree는 트레이닝 데이터 일부에 Overfitting하는 경향을 가짐
2) 서로 다른 방향으로 Overfitting된 Tree를 많이 만들면 그 결과를 평균냄으로써 Variance를 줄일 수 있음!!

   (Tree Model의 예측 성능이 유지되면서 Overfitting이 줄어드는 것이 수학적으로 이미 증명됨)
3) Tree의 개수 K가 클 수록 예측 결과가 좋아짐 but 연산량 ↑

RandomForest

Hyper Parameter

n_estimators : Tree의 개수 K

max_depth : Tree의 max_depth

min_samples_split : 노드를 분할하기 위한 최소한의 샘플 데이터수 -> 작게 설정할수록 분할 노드가 많아지므로 과적합 가능성 증가

min_samples_leaf : 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수

max_features : 선택 feature의 개수

.

.

.