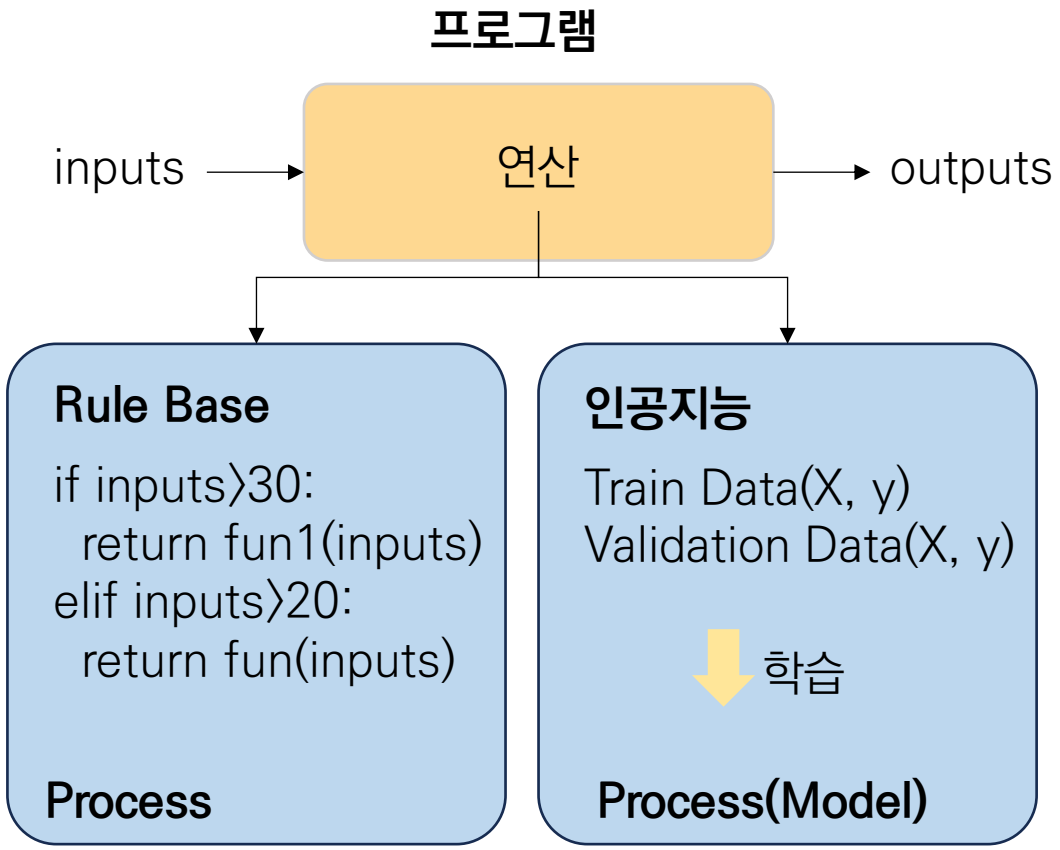


AI : 학습 데이터를
기준으로 함수(program)을 만드는 과정

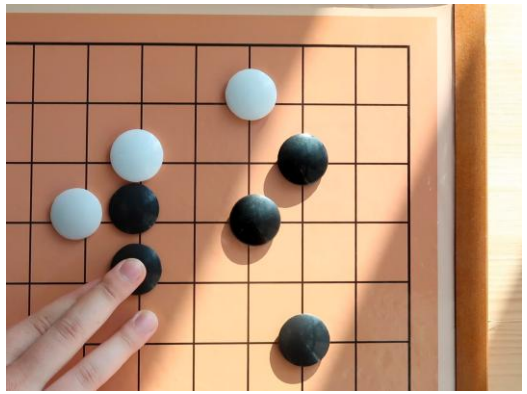
지도학습

분류 : 개, 고양이, 원숭이 처럼 클래스 분류
회귀 : 숫자 분류

비지도학습

군집 : 독립변수를 기반으로 k개 묶음
차원축소 : 컬럼이 5개면 3개로 축소
연관규칙 : 데이터 항목끼리 함께 등장하는 경향성을 찾는 알고리즘

강화학습



지도학습
- 딥러닝을 제외한 부분

각각을 독립변수라 부름

지점	시간	기온(℃)	풍속(m/s)	풍향(16방위)	습도(%)	증기압(hPa)
속초(90)	2025-06-20 01:00	23.8	1.3	50	77	22.7
속초(90)	2025-06-20 02:00	23.9	1	180	69	20.4
속초(90)	2025-06-20 03:00	24.9	1.4	360	66	20.8
속초(90)	2025-06-20 04:00	25.5	1.8	50	67	21.8
속초(90)	2025-06-20 05:00	25.5	1	160	70	22.8
속초(90)	2025-06-20 06:00	26.1	2.4	160	68	22.9
속초(90)	2025-06-20 07:00	24.5	2.9	160	75	22.9
속초(90)	2025-06-20 08:00	23.3	2.4	160	85	24.3
속초(90)	2025-06-20 09:00	23.9	1	200	90	26.6
속초(90)	2025-06-20 10:00	25.7	3.7	160	74	24.4

종속변수

시간	강수량(mm)
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	0
2025-06-20 05:00	
2025-06-20 06:00	0
2025-06-20 07:00	0.2
2025-06-20 08:00	0.4
2025-06-20 09:00	0.3
2025-06-20 10:00	0

종속변수

강수유무
비안옴(0)
비안옴(0)
비옴(1)
비옴(1)
비옴(1)
비안옴(0)

종속변수가 실수를 예측한다면? -> 회귀
종속변수가 분류를 한다면? -> 분류

AI도 숫자로 연산을 해야되서 분류도 0, 1로 코딩 되어야 함

비가 오는지를 모델이 맞추고 싶다면?
> 비온 값을 1로 안온 값을 0으로
비가 안오는 날을 맞추고 싶다면?
> 비가 안온 값을 1로 비가 온 값을 0으로

학습자료

각각을 독립변수라 부름

종속변수

지점	시간	기온(℃)	풍속(m/s)	풍향(16방위)	습도(%)	증기압(hPa)
속초(90)	2025-06-20 01:00	23.8	1.3	50	77	22.7
속초(90)	2025-06-20 02:00	23.9	1	180	69	20.4
속초(90)	2025-06-20 03:00	24.9	1.4	360	66	20.8
속초(90)	2025-06-20 04:00	25.5	1.8	50	67	21.8
속초(90)	2025-06-20 05:00	25.5	1	160	70	22.8
속초(90)	2025-06-20 06:00	26.1	2.4	160	68	22.9
속초(90)	2025-06-20 07:00	24.5	2.9	160	75	22.9
속초(90)	2025-06-20 08:00	23.3	2.4	160	85	24.3
속초(90)	2025-06-20 09:00	23.9	1	200	90	26.6
속초(90)	2025-06-20 10:00	25.7	3.7	160	74	24.4

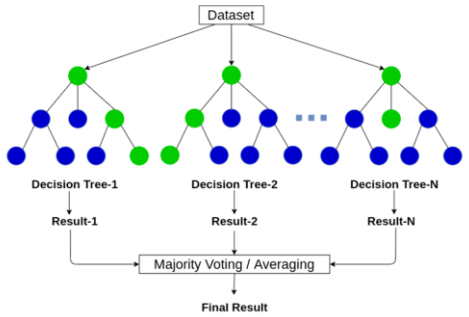
시간	강수량(mm)
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	0
2025-06-20 05:00	
2025-06-20 06:00	0
2025-06-20 07:00	0.2
2025-06-20 08:00	0.4
2025-06-20 09:00	0.3
2025-06-20 10:00	0

학습 ← 어떤 모델로??

선형회귀 SVM XGBoost Random Forest LightGBM 딥러닝

기온, 풍속, 풍향, 습도, 증기압을 넣으면 강수량이 나오는 프로세스 생성

Random Forest



기온, 풍속, 풍향, 습도, 증기압을 넣으면 강수량이 나오는 프로세스 생성

어느정도 성능이 나오는지 평가가 필요

학습에 사용한 자료는 당연히 잘 맞출 거니까
학습에 사용하지 않은 자료로 평가

Test 자료

독립변수

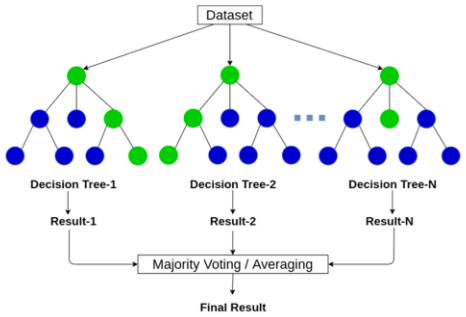
지점	시간	기온(℃)	풍속(m/s)	풍향(16방위)	습도(%)	증기압(hPa)
속초(90)	2025-06-20 01:00	23.8	1.3	50	77	22.7
속초(90)	2025-06-20 02:00	23.9	1	180	69	20.4
속초(90)	2025-06-20 03:00	24.9	1.4	360	66	20.8
속초(90)	2025-06-20 04:00	25.5	1.8	50	67	21.8
속초(90)	2025-06-20 05:00	25.5	1	160	70	22.8
속초(90)	2025-06-20 06:00	26.1	2.4	160	68	22.9
속초(90)	2025-06-20 07:00	24.5	2.9	160	75	22.9
속초(90)	2025-06-20 08:00	23.3	2.4	160	85	24.3
속초(90)	2025-06-20 09:00	23.9	1	200	90	26.6
속초(90)	2025-06-20 10:00	25.7		160	74	24.4

학습한 적 없는 2025-06-22 라 가정

종속변수

시간	강수량(mm)
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	0
2025-06-20 05:00	
2025-06-20 06:00	0
2025-06-20 07:00	0.2
2025-06-20 08:00	0.4
2025-06-20 09:00	0.3
2025-06-20 10:00	0

Random Forest



예측값

시간	강수량(mm)
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	0.3
2025-06-20 05:00	
2025-06-20 06:00	0.5
2025-06-20 07:00	0.0
2025-06-20 08:00	1.2
2025-06-20 09:00	0.6
2025-06-20 10:00	0.0

Test 자료

독립변수

지점	시간	기온(℃)	풍속(m/s)	풍향(16방위)	습도(%)	증기압(hPa)
속초(90)	2025-06-20 01:00	23.8	1.3	50	77	22.7
속초(90)	2025-06-20 02:00	23.9	1	180	69	20.4
속초(90)	2025-06-20 03:00	24.9	1.4	360	66	20.8
속초(90)	2025-06-20 04:00	25.5	1.8	50	67	21.8
속초(90)	2025-06-20 05:00	25.5	1	160	70	22.8
속초(90)	2025-06-20 06:00	26.1	2.4	160	68	22.9
속초(90)	2025-06-20 07:00	24.5	2.9	160	75	22.9
속초(90)	2025-06-20 08:00	23.3	2.4	160	85	24.3
속초(90)	2025-06-20 09:00	23.9	1	200	90	26.6
속초(90)	2025-06-20 10:00	25.7		160	74	24.4

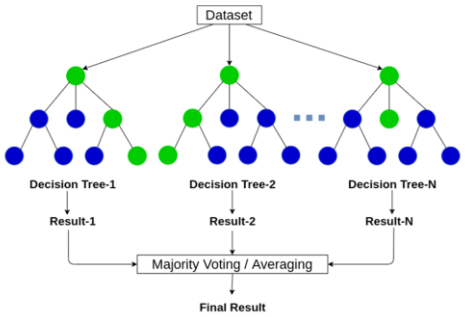
학습한 적 없는 2025-06-22 라 가정

종속변수

시간	강수량(mm)
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	0
2025-06-20 05:00	
2025-06-20 06:00	0
2025-06-20 07:00	0.2
2025-06-20 08:00	0.4
2025-06-20 09:00	0.3
2025-06-20 10:00	0

학습한 적 없는 2025-06-22 라 가정

Random Forest



예측값

시간	강수량(mm)
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	0.3
2025-06-20 05:00	
2025-06-20 06:00	0.5
2025-06-20 07:00	0.0
2025-06-20 08:00	1.2
2025-06-20 09:00	0.6
2025-06-20 10:00	0.0

학습한 적 없는 2025-06-22 라 가정

성능평가 지표로 평가

함수	설명
평균 제곱 오차(MSE)	$\sum_{i=1}^{n}(y_i-\hat{y}_i)^2/n$
평균 제곱근 오차(RMSE)	$\sqrt{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2/n}$
평균 절대 오차(MAE)	$\sum_{i=1}^{n}\lvert(y_i-\hat{y}_i)\rvert/n$
결정계수(R^2)	$R^2=1-\frac{\sum(y_i-\hat{y}_i)^2}{\sum(y_i-\bar{y})^2}=\frac{SSR}{SST}$
수정결정계수($R^2{}_{adj}$)	$R^2_{adj}=1-\left[\frac{(1-R^2)(n-1)}{n-p-1}\right]$
MAPE	$\sum_{i=1}^{n}\lvert(y_i-\hat{y}_i)/y_i\rvert\times 100/n$
Huber Loss	임계값 이하 MAE 임계값 이상 MSE

오차제곱합 = (0−0.3)^2 + (0−0.5)^2 + … + (0−0)^2
RMSE = 오차제곱합/자료개수

절대오차 = |0−0.3| + |0−0.5| + … + |0−0|
MAE = 절대오차/자료개수
MAE가 3이면 평균적으로 3정도 차이가 난다.
해석 가능

강수 유무 분류라 가정

Test 자료

독립변수

지점	시간	기온(℃)	풍속(m/s)	풍향(16방위)	습도(%)	증기압(hPa)
속초(90)	2025-06-20 01:00	23.8	1.3	50	77	22.7
속초(90)	2025-06-20 02:00	23.9	1	180	69	20.4
속초(90)	2025-06-20 03:00	24.9	1.4	360	66	20.8
속초(90)	2025-06-20 04:00	25.5	1.8	50	67	21.8
속초(90)	2025-06-20 05:00	25.5	1	160	70	22.8
속초(90)	2025-06-20 06:00	26.1	2.4	160	68	22.9
속초(90)	2025-06-20 07:00	24.5	2.9	160	75	22.9
속초(90)	2025-06-20 08:00	23.3	2.4	160	85	24.3
속초(90)	2025-06-20 09:00	23.9	1	200	90	26.6
속초(90)	2025-06-20 10:00	25.7		160	74	24.4

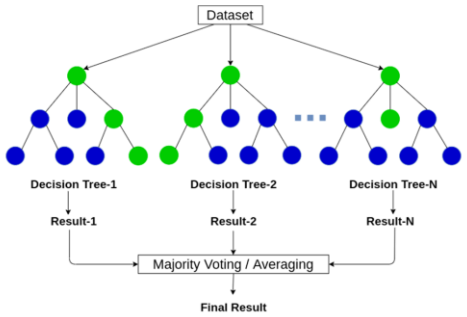
학습한 적 없는 2025-06-22 라 가정

종속변수

시간	강수유무
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	비안옴(0)
2025-06-20 05:00	
2025-06-20 06:00	비안옴(0)
2025-06-20 07:00	비옴(1)
2025-06-20 08:00	비옴(1)
2025-06-20 09:00	비옴(1)
2025-06-20 10:00	비안옴(0)

학습한 적 없는 2025-06-22 라 가정

Random Forest



예측값

시간	강수유무
2025-06-20 01:00	
2025-06-20 02:00	
2025-06-20 03:00	
2025-06-20 04:00	비안옴(0)
2025-06-20 05:00	
2025-06-20 06:00	비안옴(0)
2025-06-20 07:00	비안옴(0)
2025-06-20 08:00	비안옴(0)
2025-06-20 09:00	비안옴(0)
2025-06-20 10:00	비안옴(0)

학습한 적 없는 2025-06-22 라 가정

구분		실제값	
		True	False
예측값	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

구분		실제값	
		True	False
예측값	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

분류지표	수식
정확도(Accuracy)	(TP+TN)/(TP+FP+FN+TN)
정밀도(Precision)	TP/(TP+FP)
민감도/재현율 (Sensitivity/Recall)	TP/(TP+FN)
특이도(Specificity)	TN/(TN+FP)
F1 Score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$

TP : 실제 값과 모델 예측 값이 1인 경우
FP : 실제 값은 0인데 모델은 1로 예측한 경우
FN : 실제 값은 1인데 모델은 0으로 예측한 경우
TN : 실제 값과 모델 예측 값이 모두 0인 경우

정확도: 전체 자료 중에 모델이 맞춘 개수

정밀도: 모델이 1이라 예측한 것 중 실제 값이 1인 비율

민감도: 실제 1인 값 중 모델이 1이라 예측한 것의 비율

특이도: 실제 0인 값 중 모델이 0이라 예측한 것의 비율

F1 Score 는 민감도와 특이도의 조화 평균

1종 오류

구분		실제값	
		True	False
예측값	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

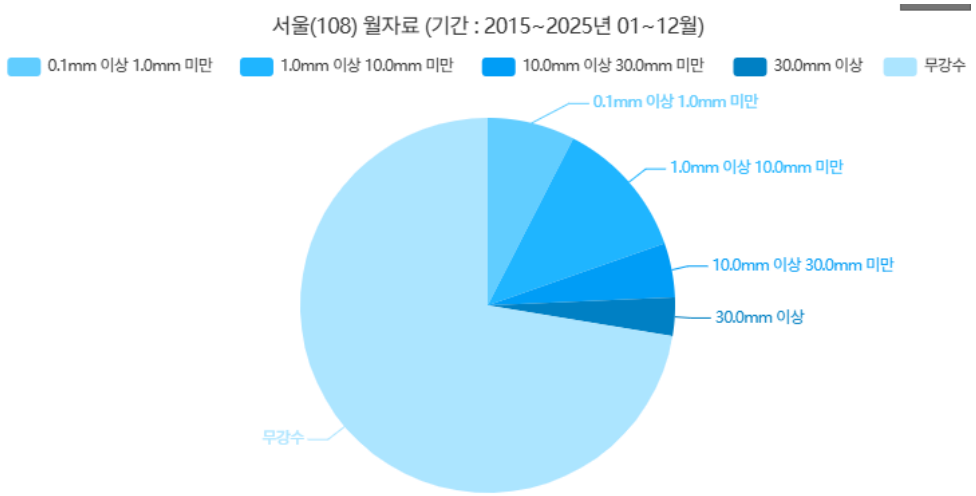
2종 오류

분류지표	수식
정확도(Accuracy)	(TP+TN)/(TP+FP+FN+TN)
정밀도(Precision)	TP/(TP+FP)
민감도/재현율 (Sensitivity/Recall)	TP/(TP+FN)
특이도(Specificity)	TN/(TN+FP)
F1 Score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$

1종오류 – 비와요 라고 예보 했는데 비 안 오는 경우
2종오류 – 비안와요라고 예보 했는데 비 오는 경우

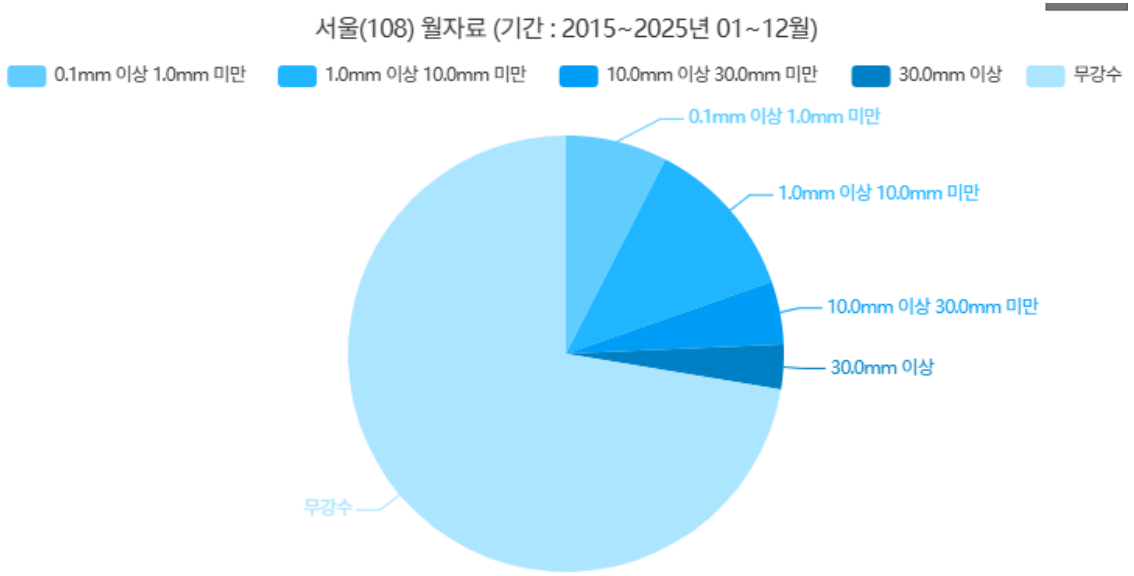
비 안오는 날이 월등히 많음
모델은 비 안온다고만 예측해도 정확도 기준으로는 매우 높은 값을 가짐

1종오류 – 암이에요 라고 했는데 암이 아닌 경우
2종오류 – 암이 아니에요라고 진단했는데 암인 경우



분류지표	수식
정확도(Accuracy)	(TP+TN)/(TP+FP+FN+TN)
정밀도(Precision)	TP/(TP+FP)
민감도/재현율 (Sensitivity/Recall)	TP/(TP+FN)
특이도(Specificity)	TN/(TN+FP)
F1 Score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$

구분		실제값	
		True	False
예측값	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)



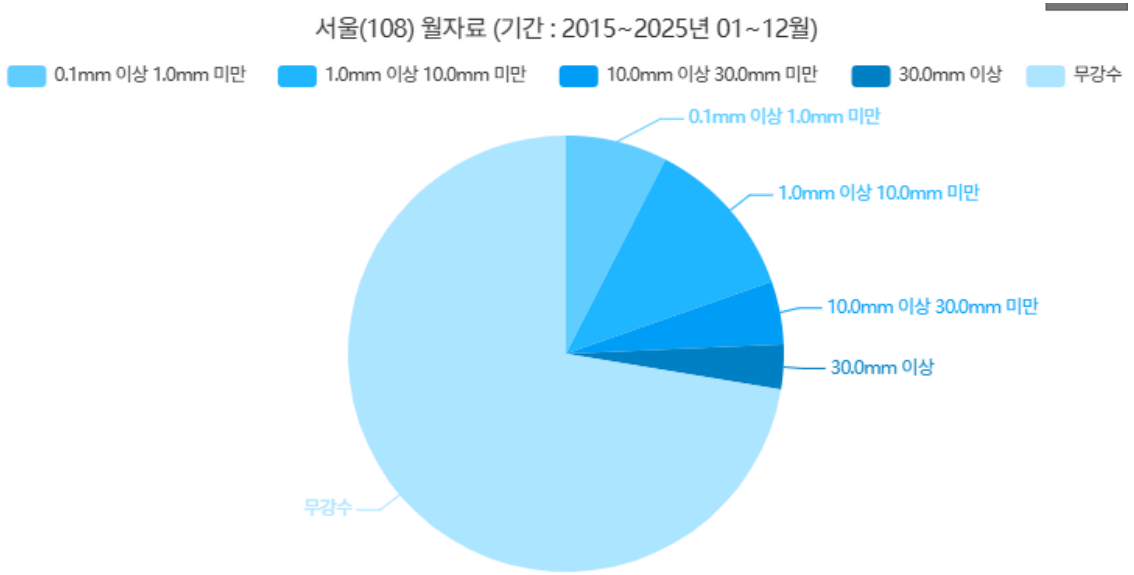
비 안 오는 날이 월등히 많음
모델은 비 안 온다고만 예측해도 정확도 기준으로는
매우 높은 값을 가짐

비 온다고 한적이 없으니 정밀도, 민감도는 0

그럼 데이터 불균형 하면 정밀도, 민감도를 보면 되는건가?

분류지표	수식
정확도(Accuracy)	(TP+TN)/(TP+FP+FN+TN)
정밀도(Precision)	TP/(TP+FP)
민감도/재현율 (Sensitivity/Recall)	TP/(TP+FN)
특이도(Specificity)	TN/(TN+FP)
F1 Score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$

구분		실제값	
		True	False
예측값	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)



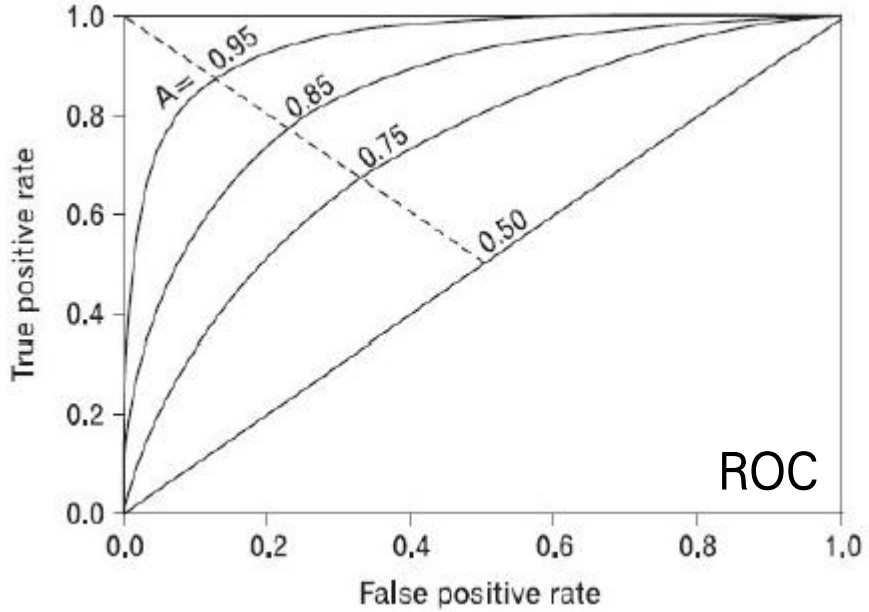
그럼 데이터 불균형 하면 정밀도, 민감도를 보면 되는 건가?

반대로 **비 안온 날**을 1로 뒀다 가정
실제 **비 안온 날** 중 온다고 예측한 적이 없기 때문에 FN이 0
즉, 민감도는 1
특이도는 실제 비가 온 날 중 비가 안 온다고 한 값들이 많기 때문에 1보다 작은 값을 가짐

민감도를 지표로 봤다면 성능이 안 좋은데 좋다고 한거니 문제가 발생

그래서 민감도랑 특이도 둘 다 고려할 수 있는 F1 Score를 쓰게 됨

민감도



1-특이도

분류의 기준은 왜 0.5여야 돼?

Threshold 를 바꿔가면서
민감도랑 특이도 정밀도를 평가

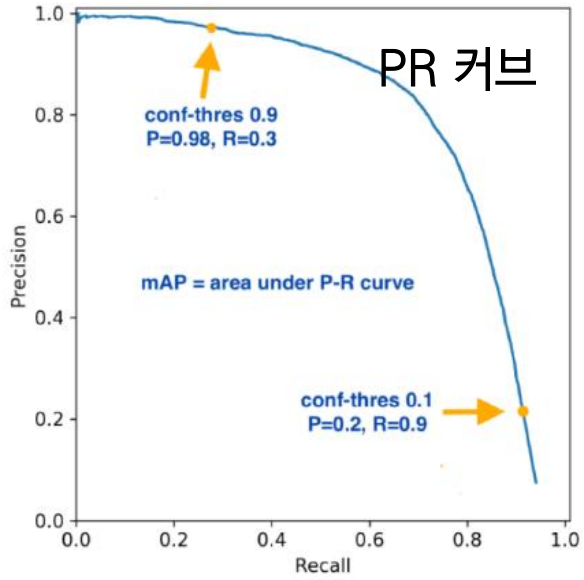
그러면 ROC, PR 커브가 나옴

대충 생각해보면,

이 모델은 51%확률로 비가 온다고
했었어도 잘 맞아

이런 느낌

정밀도



민감도

프로그램

inputs → 연산 → outputs

인공지능

Train Data(X, y)
Validation Data(X, y)

↓ 학습

Process(Model)

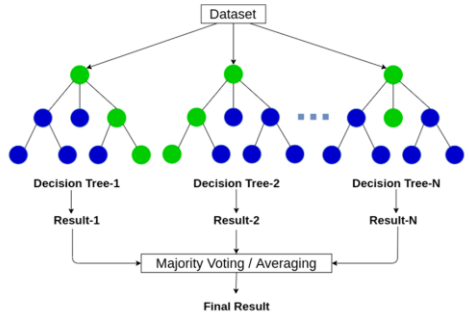
Train Data → Model 생성 → Test Data → 예측값

↓

성능평가

Validation Data는??

Random Forest



API Reference > sklearn.ensemble > RandomForestRegressor

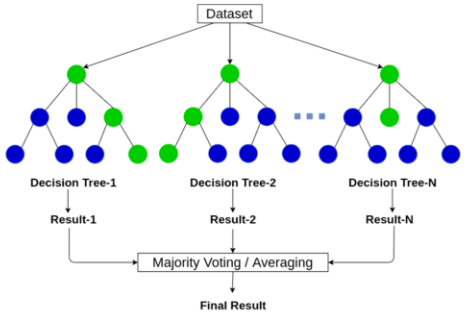
RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *,
criterion='squared_error', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=1.0, max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None,
random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None,
monotonic_cst=None)                                                    [source]
```

이 수많은 옵션 중 어떤 옵션을
선택해야 제일 좋은 모델이야??

Validation Data는??

Random Forest



함수의 파라미터들을 테스트 해봐야 되는데,
Train 데이터 기준으로 옵션을 평가 하자고 하니
학습을 Train데이터로 했으니 무조건 다 잘 나올 거 같고

Test 데이터로 평가하자고 하니
그럼 평가하는 데이터에 맞게 옵션을 설정한 게 되고

옵션을 평가할 데이터를 따로 두자!
> Validation Data

API Reference > sklearn.ensemble > RandomForestRegressor

RandomForestRegressor

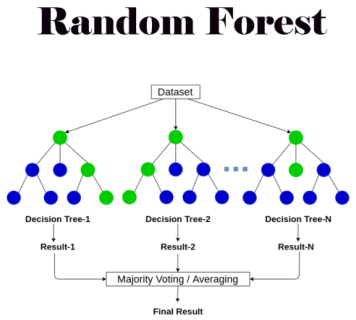
```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *,
criterion='squared_error', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=1.0, max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None,
random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None,
monotonic_cst=None)                                                    [source]
```

이 수많은 옵션 중 어떤 옵션을
선택해야 제일 좋은 모델이야??

Train

Validation

RMSE 0.5



n_estimators = 100
으로 생성한 모델

API Reference > sklearn.ensemble > RandomForestRegressor

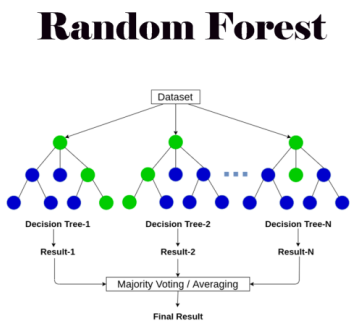
RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *,
criterion='squared_error', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=1.0, max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None,
random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None,
monotonic_cst=None)                                                    [source]
```

Train

Validation

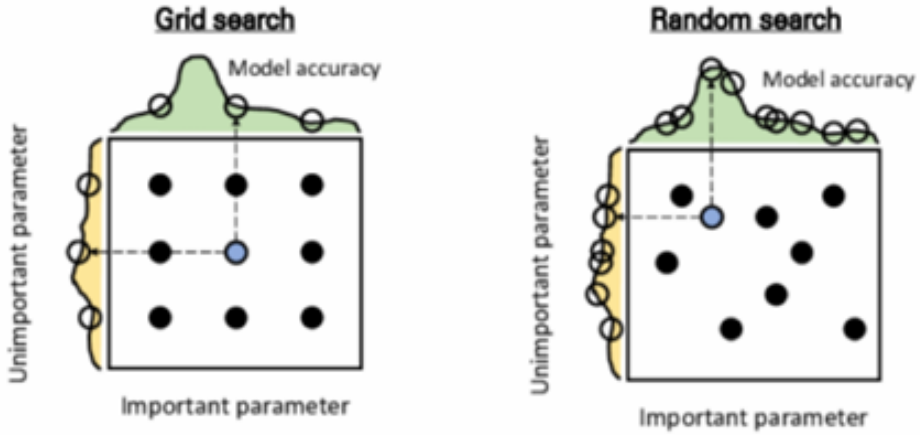
RMSE 0.3



n_estimators = 200
으로 생성한 모델

내 데이터에서는 n_estimators 가
100보다 200이 더 성능이 좋아

내 데이터의 분포에 따라
파라미터의 최적 값이 정해지므로
이런 걸 **하이퍼 파라미터**라고 함




```python
# 2. 하이퍼파라미터 후보 값 설정
param_dist = {
    'n_estimators': np.arange(10, 200, 10),  # 트리 개수 (10~190, 10 단위 증가)
    'max_depth': [None, 10, 20, 30],  # 최대 깊이
    'min_samples_split': np.arange(2, 20, 2),  # 분할을 위한 최소 샘플 수 (2~18, 2 단위 증가)
    'min_samples_leaf': np.arange(1, 10, 1),  # 리프 노드의 최소 샘플 수 (1~9)
    'bootstrap': [True, False]  # 부트스트랩 사용 여부
}

# 3. RandomForest 모델과 RandomizedSearchCV 설정
rf = RandomForestClassifier()
random_search = RandomizedSearchCV(
    rf, param_distributions=param_dist, n_iter=20, cv=3, scoring='accuracy', n_jobs=-1, random_state=42
)

# 4. 최적의 하이퍼파라미터 찾기
random_search.fit(X_train, y_train)
```

Grid : 19×4×9×9×2 =12,312회
Random : 20회

n_estimators 를 10~200까지 10 간격으로
max_depth : None, 10, 20, 30
min_samples_split 2~20까지 2간격으로
…
부트스트랩 쓴 경우 안쓴경우

모든 조합을 다 테스트해보고 그 중에 베스트를
내 하이퍼파라미터로 쓰겠다

= Grid Search

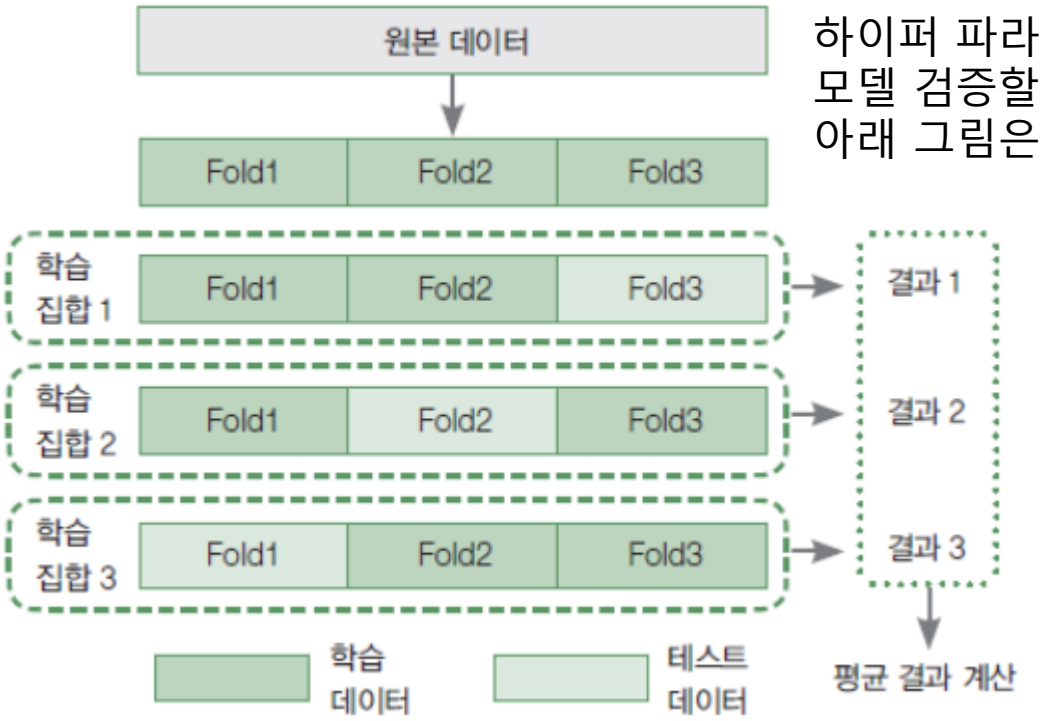
하이퍼 파라미터들 범위만 줄 테니까
n_iter 적힌 만큼만
random하게 시도해

=Random Search

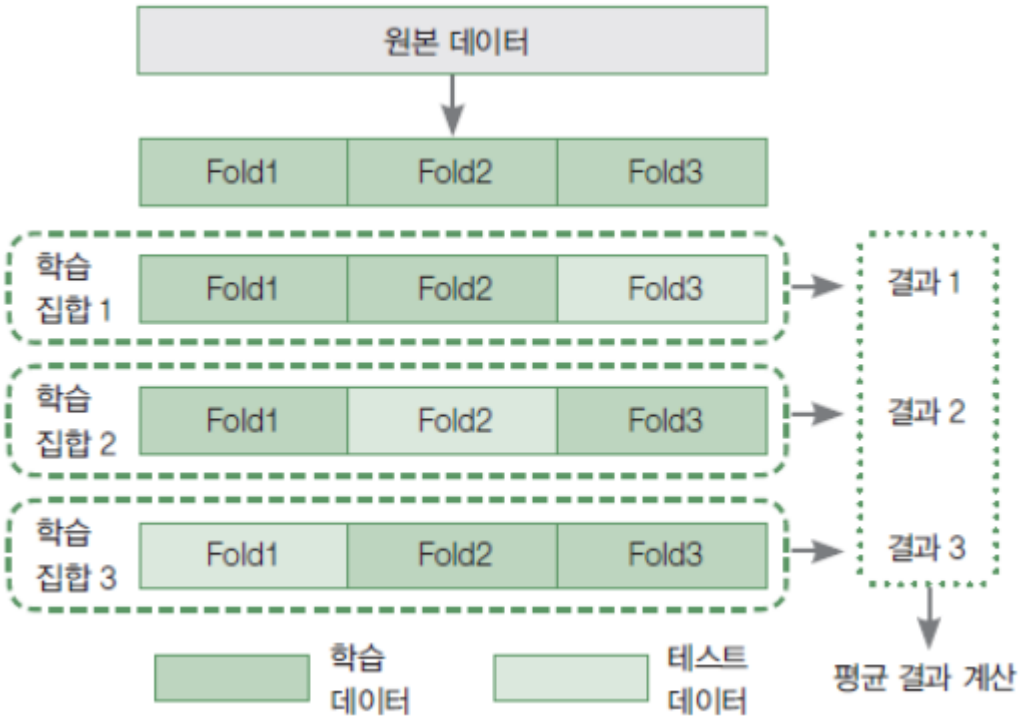
방금까지 진행한 것이 Hold Out Cross Validation 방식입니다.

Train, Valid, Test 로 데이터를 쪼개서 모델을 학습, 평가 하는 것을 의미

K Fold Cross Validation?
데이터를 K 등분 해서 하이퍼 파라미터를 튜닝하든..
평가를 하든 할 때 사용합니다.



하이퍼 파라미터 튜닝안하고 모델 검증할 때 사용한다면 아래 그림은 당연해 보임



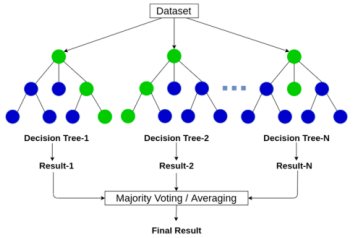
테스트 데이터라 적힌 부분을 Validation이라 생각

API Reference > sklearn.ensemble > RandomForestRegressor

RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *,
criterion='squared_error', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=1.0, max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None,
random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None,
monotonic_cst=None)                                                    [source]
```

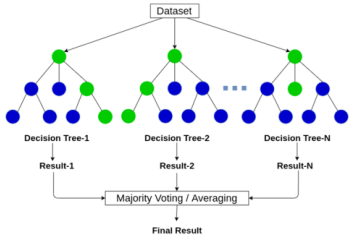
Random Forest



n_estimators = 100
으로 생성한 모델

결과1 : 0.3
결과2 : 0.2
결과3 : 0.3

Random Forest



n_estimators = 200
으로 생성한 모델

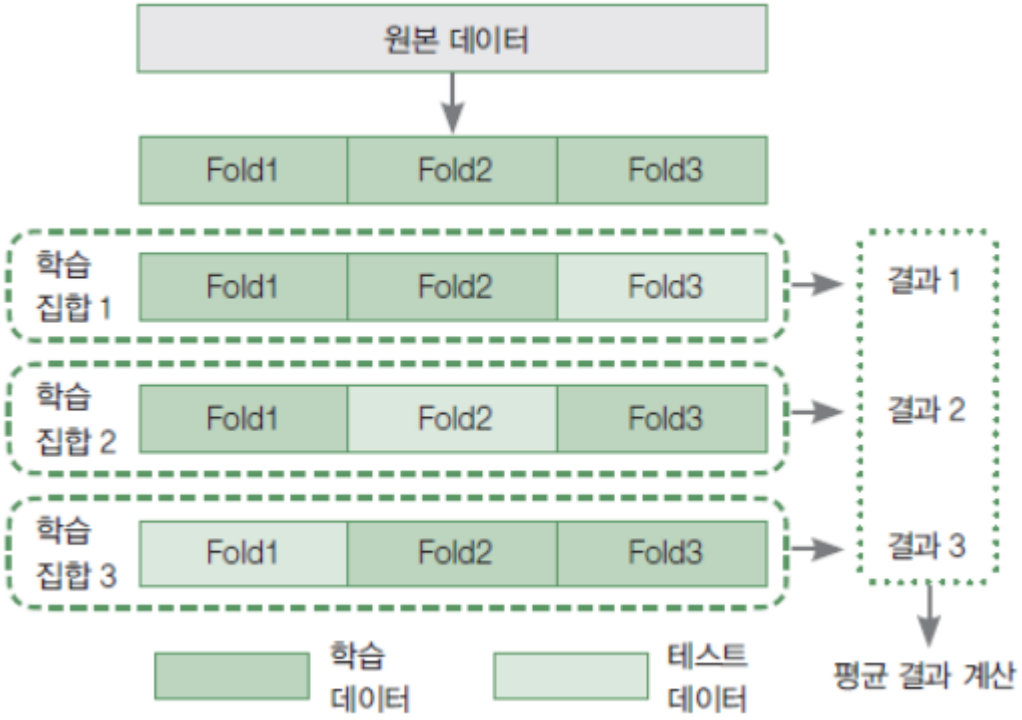
결과1: 0.2
결과2: 0.2
결과3: 0.1

어 이 모델이 더 안정적으로 좋네

Hold Out보다 조금 더 일반화가 잘된 모델을 선택 가능

> 근데 K 번 반복 해야되서 너무 오래 걸림 ...

Test 데이터를 1개씩만 두고 계속 반복 해보자
> Leave One Out Cross Validation

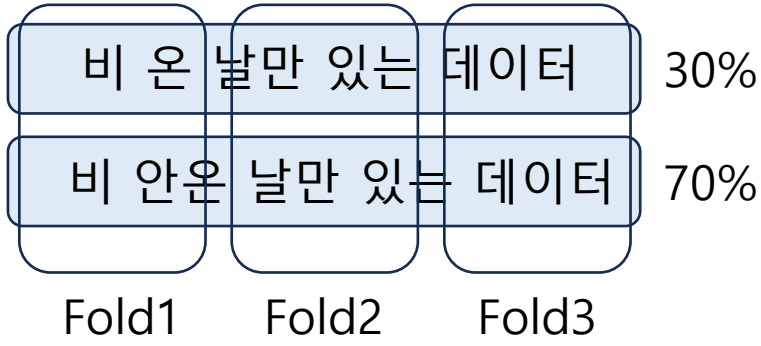


다시 강수 분류 데이터라 생각

당연히 Fold1, Fold2, Fold3
모두 강수의 비율이 같지 않음

그럼 Fold별로 비율을 맞추자!

층화추출 - 층을 나눠서 추출하자!

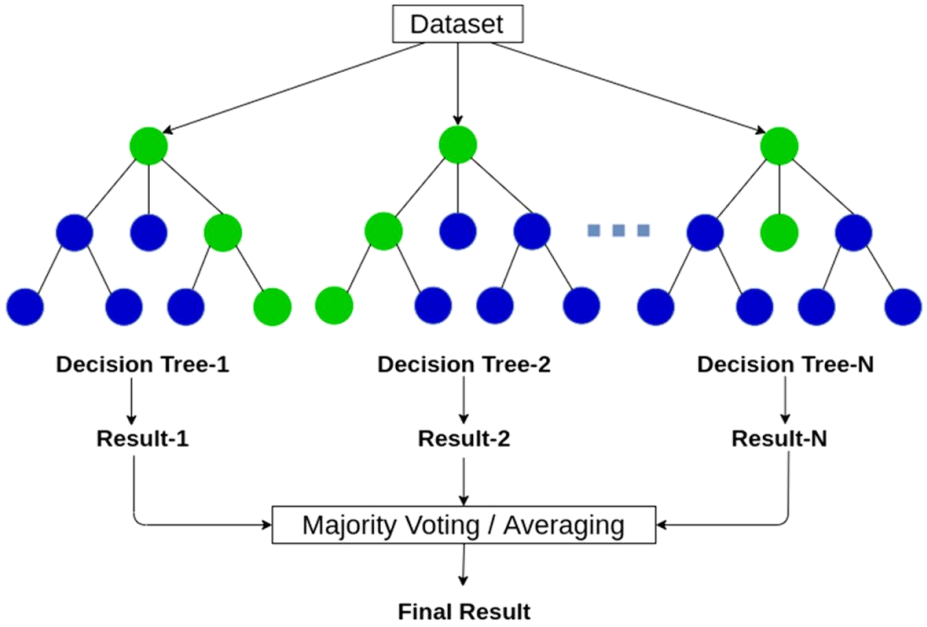


층화 추출 + k fold CV
= Stratified K Fold Cross Validation 이라 기재

앙상블 모델 설명

앙상블은 여러 모델 합쳐서
하나의 모델로 쓰겠다!

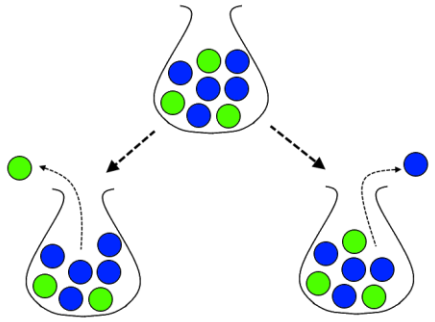
배깅
Bootstrap Aggregating
복원 추출해서 집계 하겠다!



보팅도 사실 배깅의 일종..
근데 AI 라이브러리에서 따로 구현되어 있어서
따로 얘기하심

학습 데이터 행
5,000 개 있다 가정

5000개 중 500개씩 중복을
허용해서 3회 추출
(복원 추출 == 부트스트랩)
> A, B, C 라 부르겠음



A데이터로 의사결정트리 모델 구축

B데이터로 의사결정트리 모델 구축

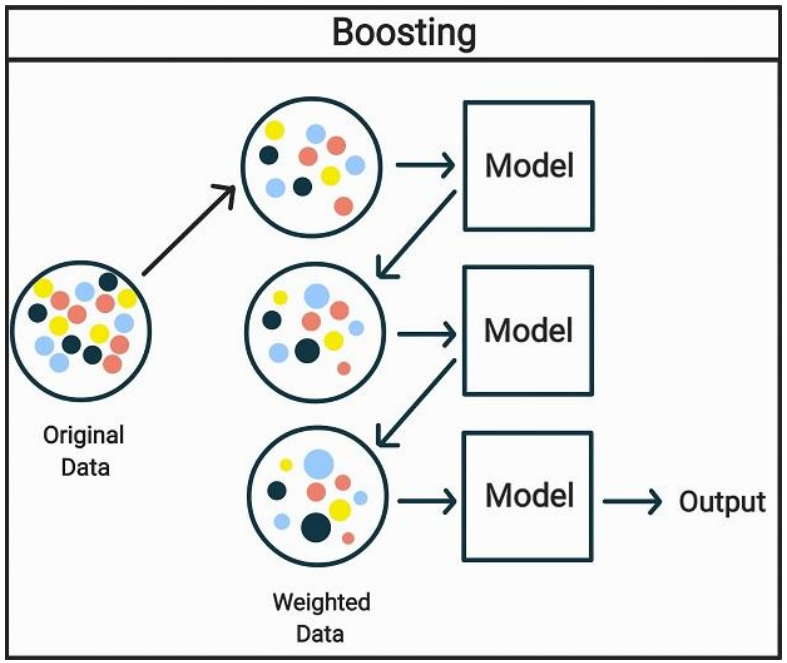
C데이터로 의사결정트리 모델 구축

여기서 나온
A, B, C 모델을
약분류기나
학습기라
부름..

모두 다른 데이터로 학습을 했으니
잘 맞추는 데이터랑 못 맞추는 데이터가 다름

이걸 민주주의처럼 투표해보자
> A는 비옴, B는 비안옴, C는 비안옴
> 비안온다는 의견이 많네 그럼 비안온다고 하자
66%로 비안와요

Boosting 모델
> 약한 학습기들을 순차적으로 학습시키며
이전 모델의 오류를 보완



학습 데이터 행
5,000 개 있다 가정

5,000개 데이터로 의사결정트리 모델(A) 구축

A 모델이 잘 못 맞추는 데이터가 더 잘 나오게
확률을 조정해서 500개 복원 추출

500개 데이터로 의사결정트리 모델(B) 구축

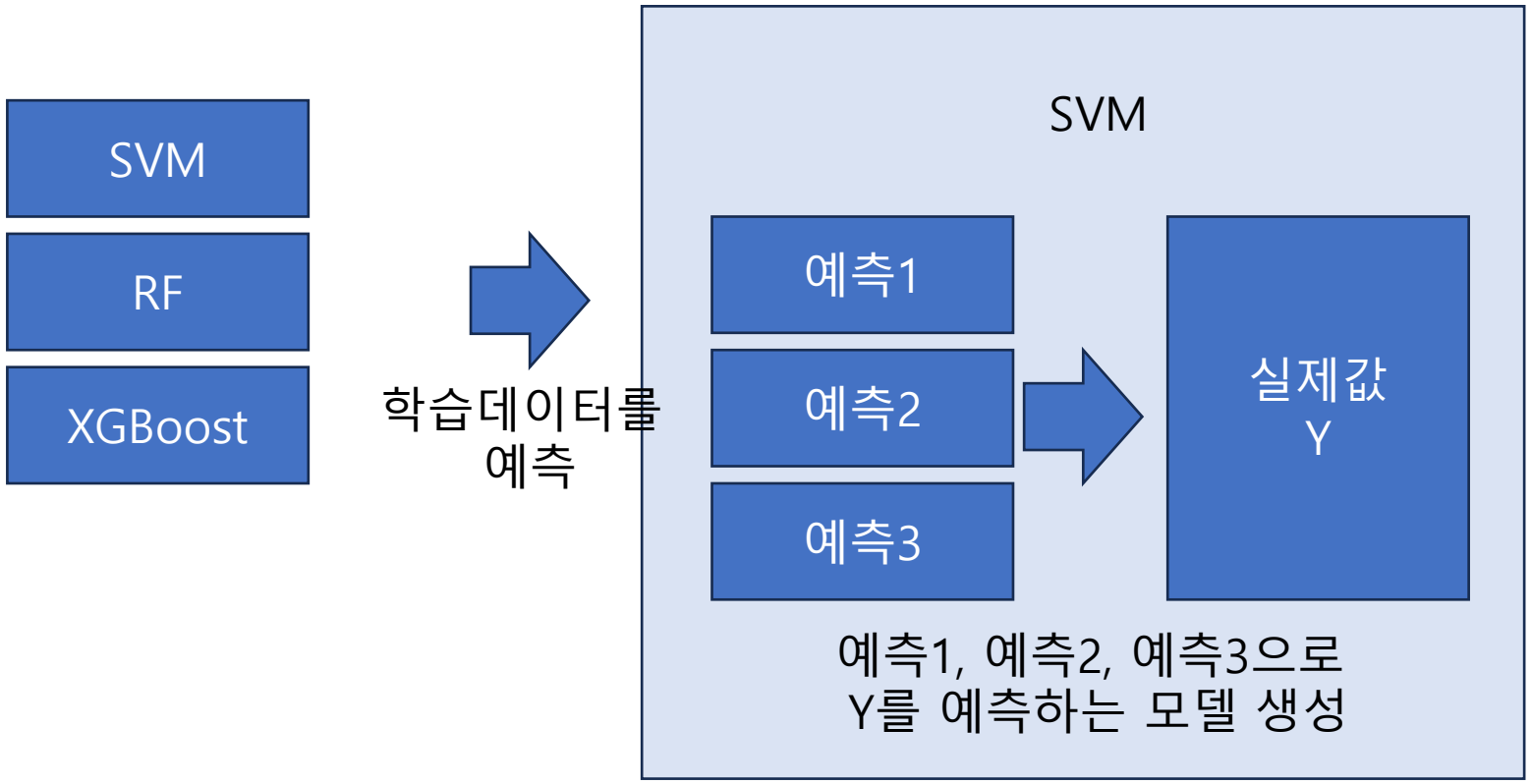
B 모델이 잘 못 맞추는 데이터가 더 잘 나오게
확률을 조정해서 500개 복원 추출

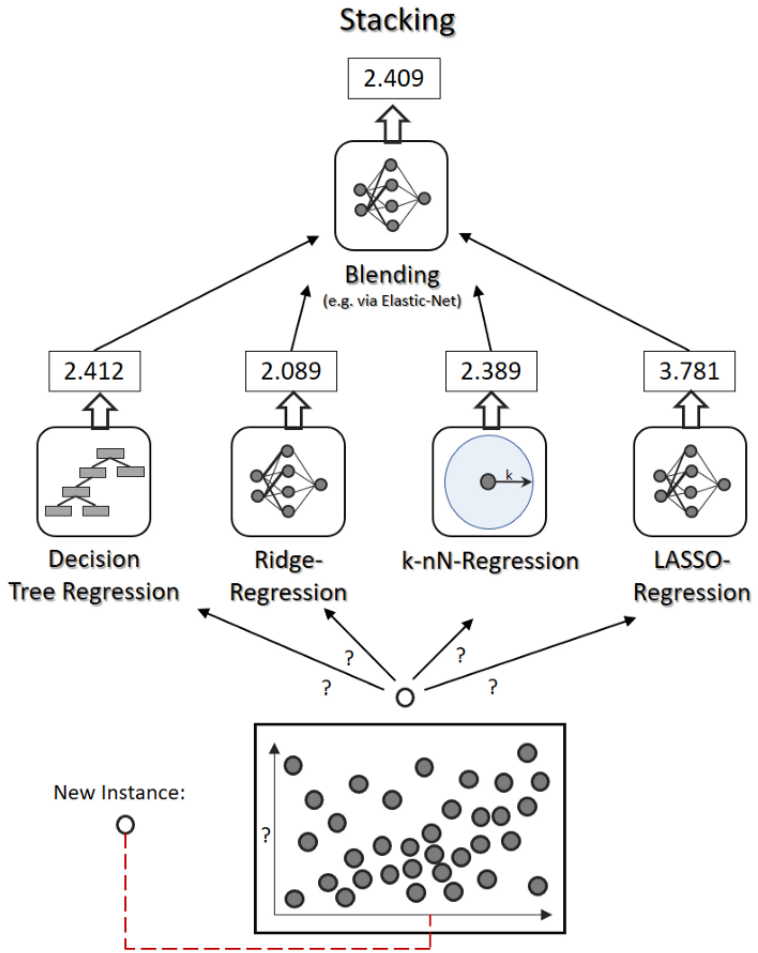
500개 데이터로 의사결정트리 모델(C) 구축

모델 A, B, C 모델을 성능에 따라
가중치를 다르게 주어 최종 결과 제시하게 함

똑똑한 사람은 투표권 3개! 아니면 1개! 이런느낌

Stacking 모델
> 모델을 2 단계로 학습





Blending 모델
➢ 모델 예측값을
가중치줘서 최종 값 예측
predict = 0.2SVM + 0.4 RF + 0.4 XGBoost
강의자료랑 내용 다름 – 확인 필요,

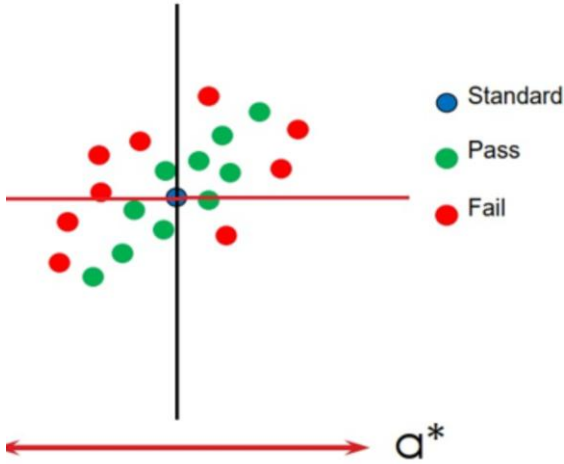
PCA 설명

500 x 3 행렬이 있다 가정

공분산이나 상관계수를 구하면 3x3으로 나옴

[1.00 0.85 -0.60
 0.85 1.00 -0.40
-0.60 -0.40 1.00]

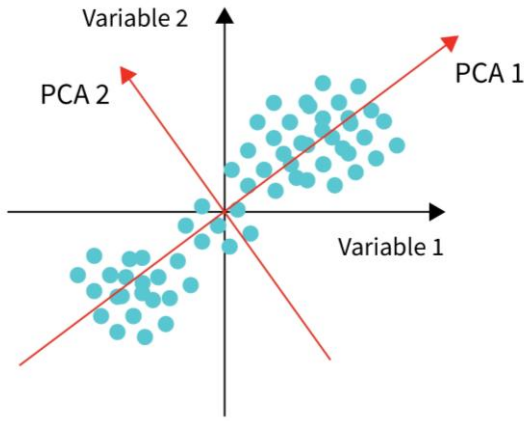
A변수와 B변수는 85% 양의 선형 상관관계가 있다.
A변수와 C변수는 60% 음의 선형 상관관계가 있다.
B변수와 C변수는 40% 음의 선형 상관관계가 있다.



행렬을 분해
 - 이 때 다양한 방식을 쓸 수 있음.
고유값 분해, 특이값 분해(SVD), UV 분해 등등

$$\lambda_1 = 2.2523, \quad \lambda_2 = 0.6313, \quad \lambda_3 = 0.1164$$

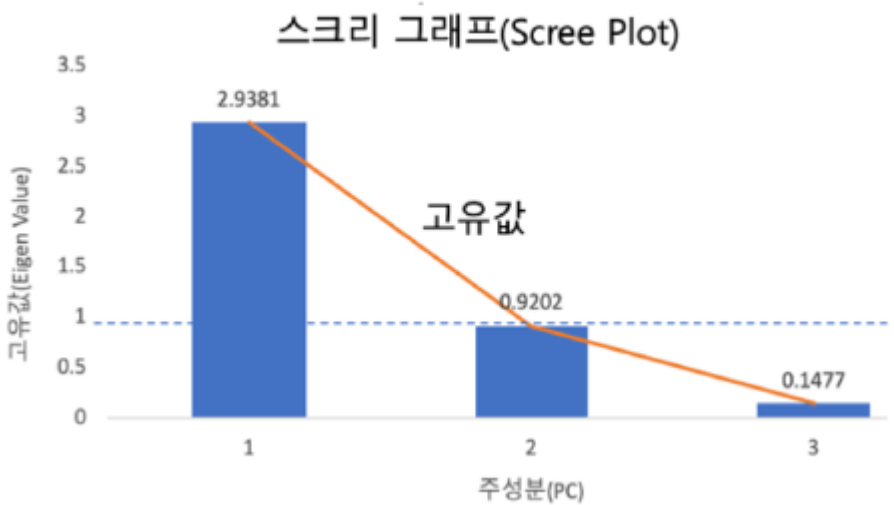
$$\mathbf{v}_1 = \begin{bmatrix} 0.638 \\ 0.591 \\ -0.494 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0.170 \\ 0.518 \\ 0.838 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 0.751 \\ -0.619 \\ 0.230 \end{bmatrix}$$



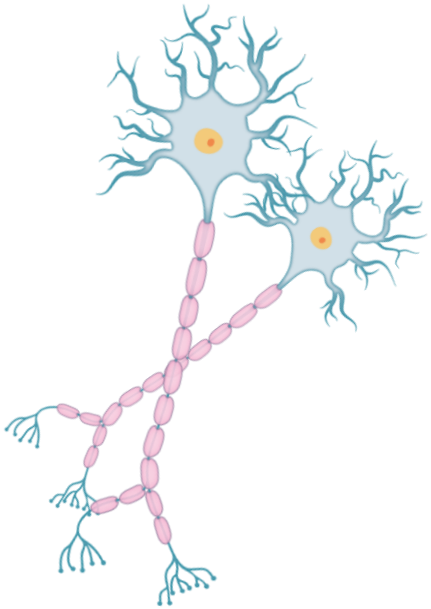
주축을 1개로 할꺼면
(500 x 3) 행렬을 v1(3 x 1)과 행렬곱 수행
(500 x 1)

주축을 2개로 할꺼면
(500 x 3) 행렬을 v1,v2 (3, 2)
(500 x 2)

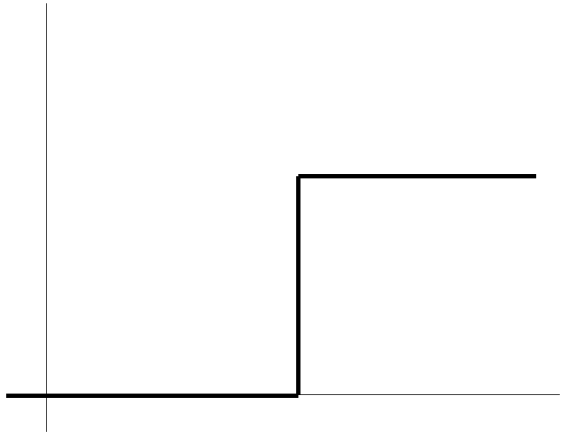
$$\lambda_1 = 2.2523, \quad \lambda_2 = 0.6313, \quad \lambda_3 = 0.1164$$



딥러닝 설명



뉴런



특정 값 이상에서 자극

큰 자극은?

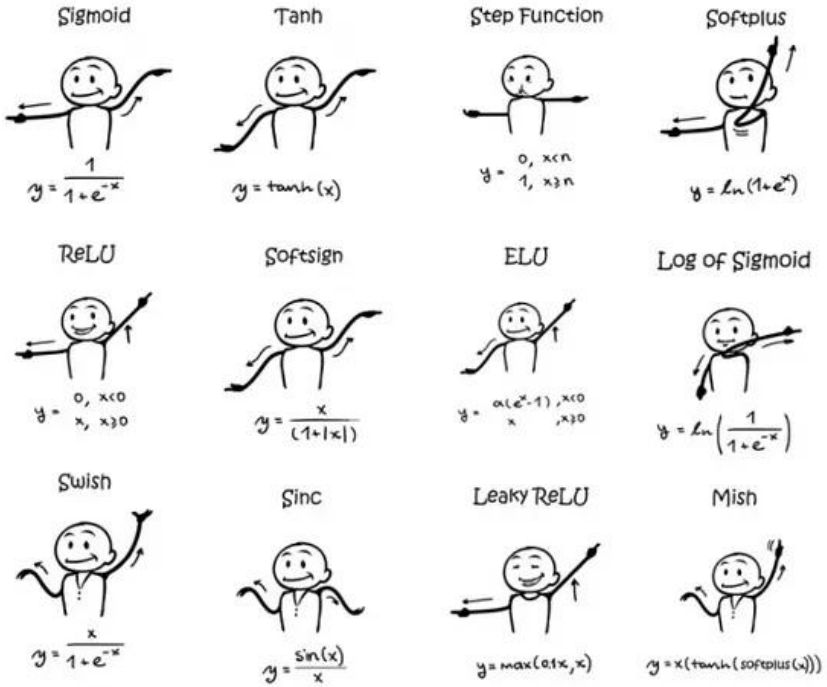
여러 뉴런이 반응(활성화)해서 크게 느낌

뉴런 1개를 따라한 모델을 퍼셉트론이라 함 xor 문제를 풀지 못함

Sign 함수는
미분 불가하니까 바꿔보자
부드러운 함수(Sigmoid) 뭐 있니?

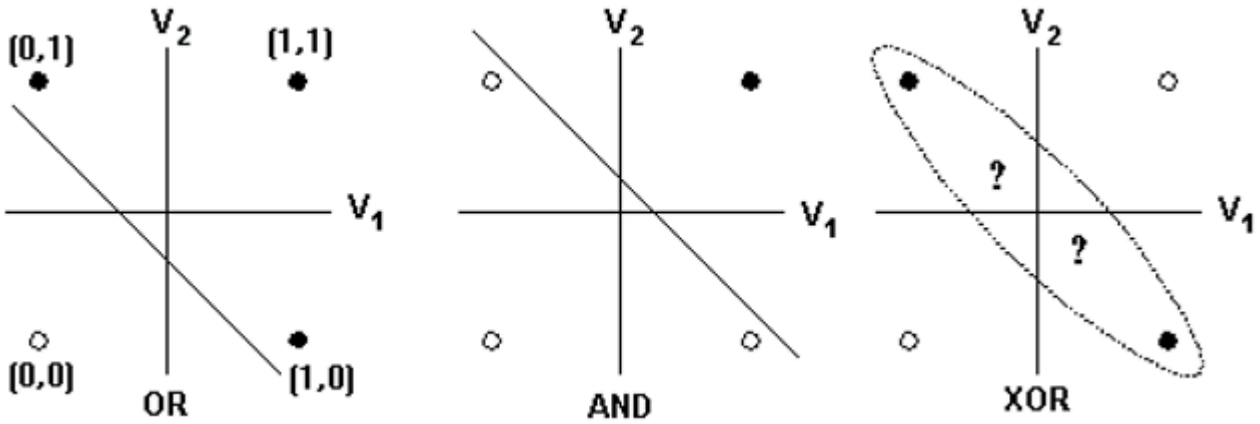
시그모이드는 부드러운 S 자 형태의
곡선을 의미(현재는
로지스틱함수로 개념이 굳음)

> active function
tanh 나 relu 같은 걸 쓰기 시작



XOR 문제

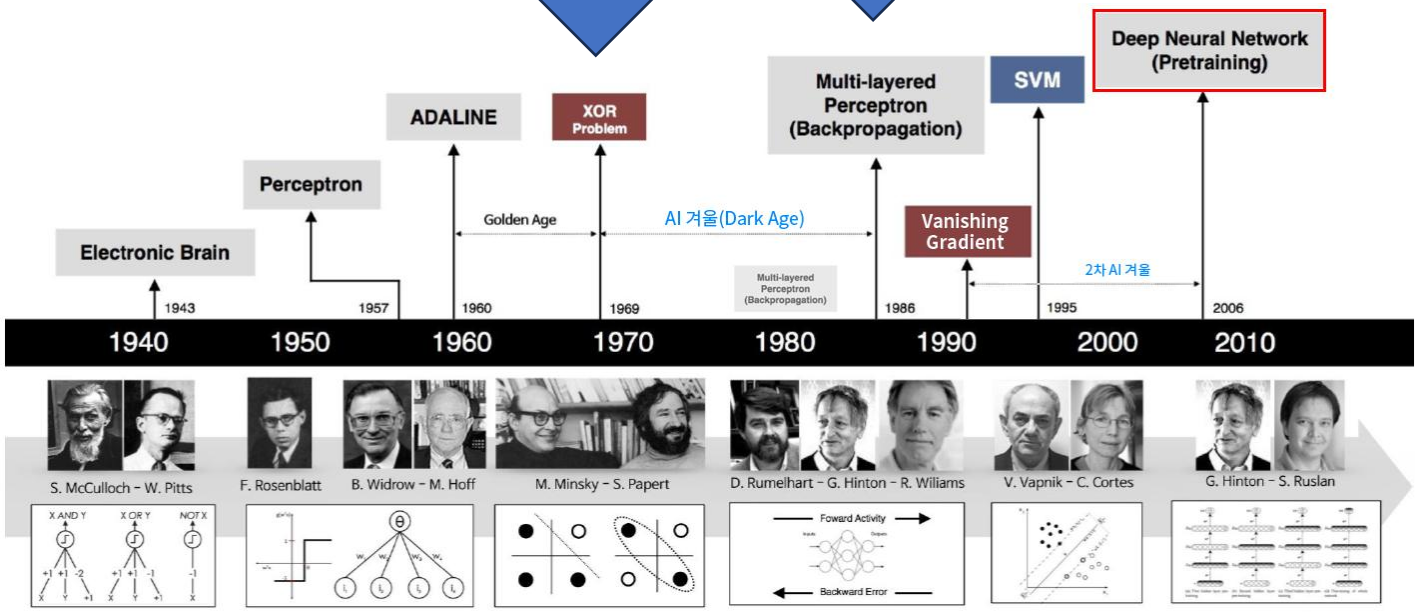
XOR 문제를 해결하지 못해 침체기가 발생
XOR문제는 아래 그림에서 검은 점과 흰 점을 구분하는 선을 찾는 문제



은닉층을
여러 개 쌓은
구조 이때 나옴

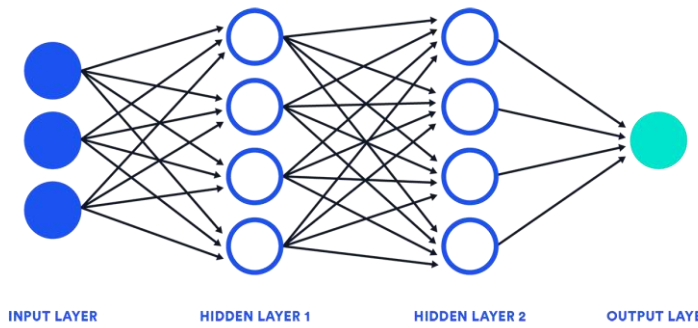
근데 SVM 나오고
성능이 SVM한테
안됐음

XOR



딥러닝 모델 구조는 아래와 같이 생김

입력층 : 들어오는 숫자 갯수
은닉층 : 중간에 연산할 노드 수
출력층 : 말그대로 나가는 숫자 갯수

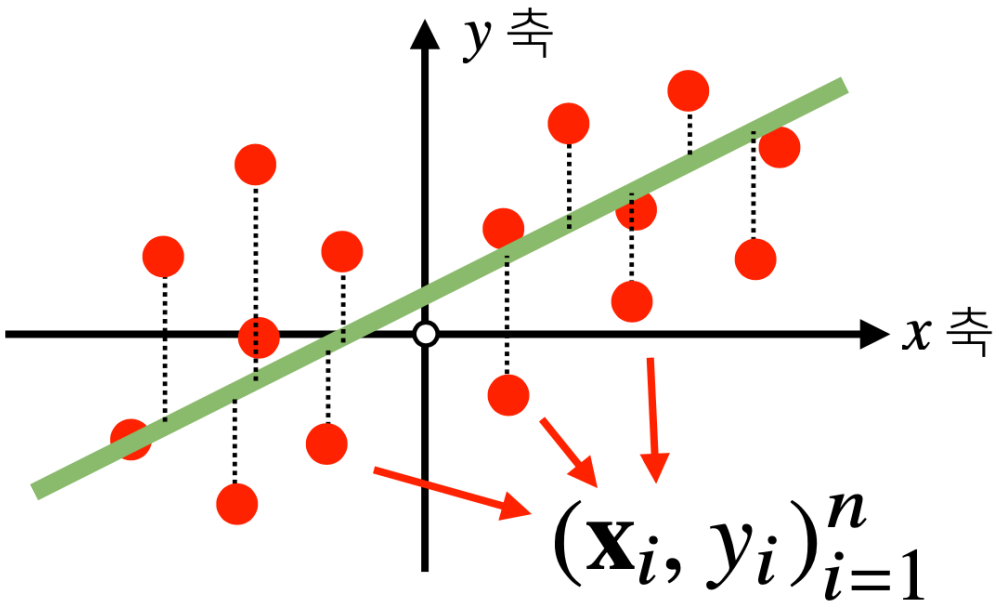


컬럼 3개를 입력받아서
연산과정을 거친뒤 4개로 출력(은닉층1)
연산과정을 거친뒤 4개로 출력(은닉층2)
4개를 입력받아서 숫자 1개로 반환

분류모델이면 0~1사이의 값
회귀 모델이면 상수

두 개 이상의 은닉층을 가질 때 부터
딥러닝이라 부름

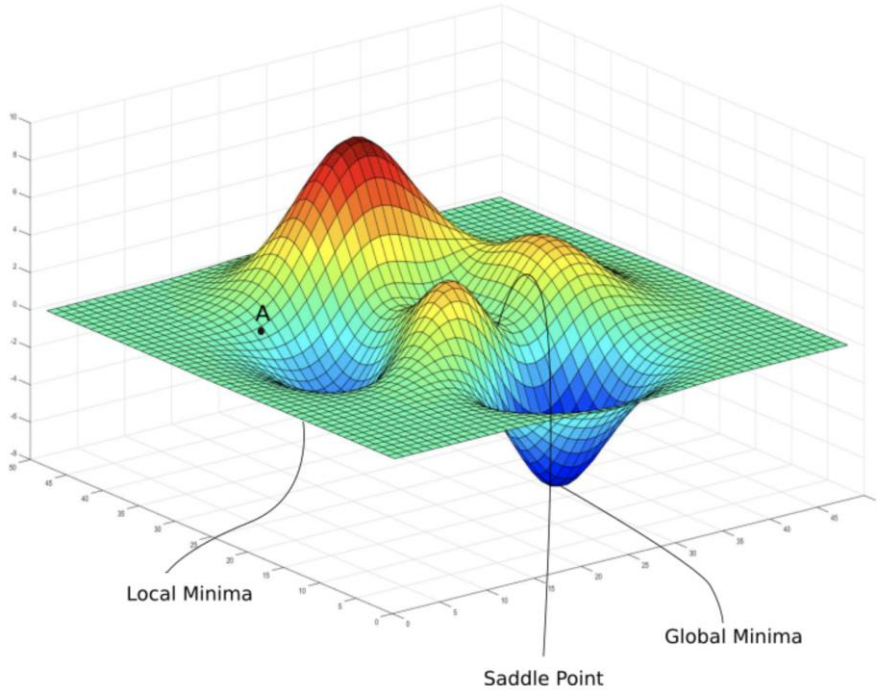
은닉층이나 출력층에서는 활성화 함수를 보통 씀
그래서 출력층에서 sigmoid 함수를 쓰면 0~1로 범위가 제한
linear 함수 쓰면 실수
relu 쓰면 0과 양의 실수로 범위가 제한됨



선형 회귀 모델은
데이터를 기준으로 오차가 최소가 되게 하는 직선을
찾는 알고리즘

우리가 최종적으로 줄이고 싶어하는 오차라는 지표를
Cost function(목적함수) 라고 함

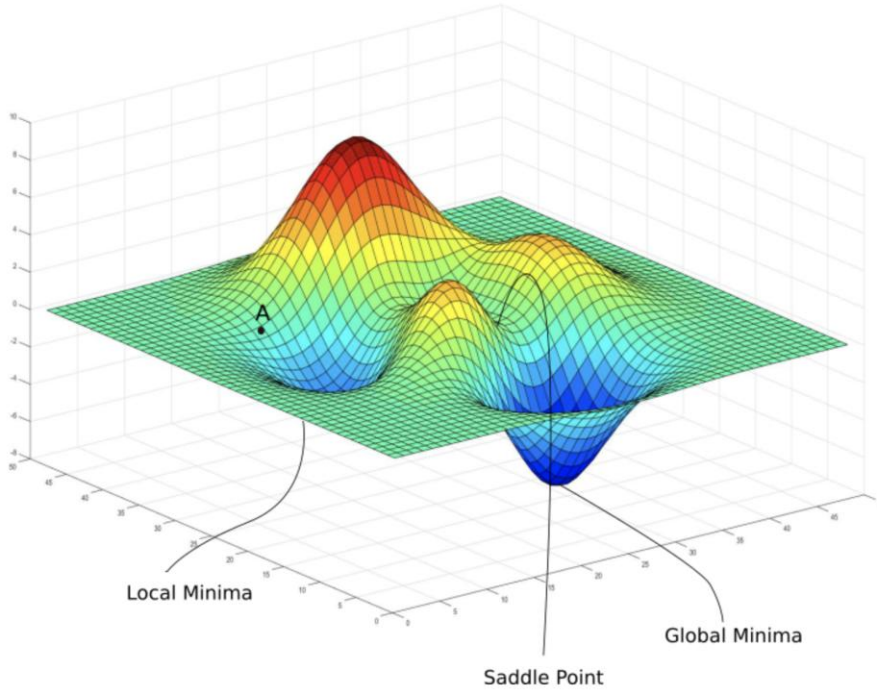
앞에서 머신러닝은
한번에 전체 데이터를 학습 할 수 있음
근데 데이터가 커서 딥러닝은 그게 안됨

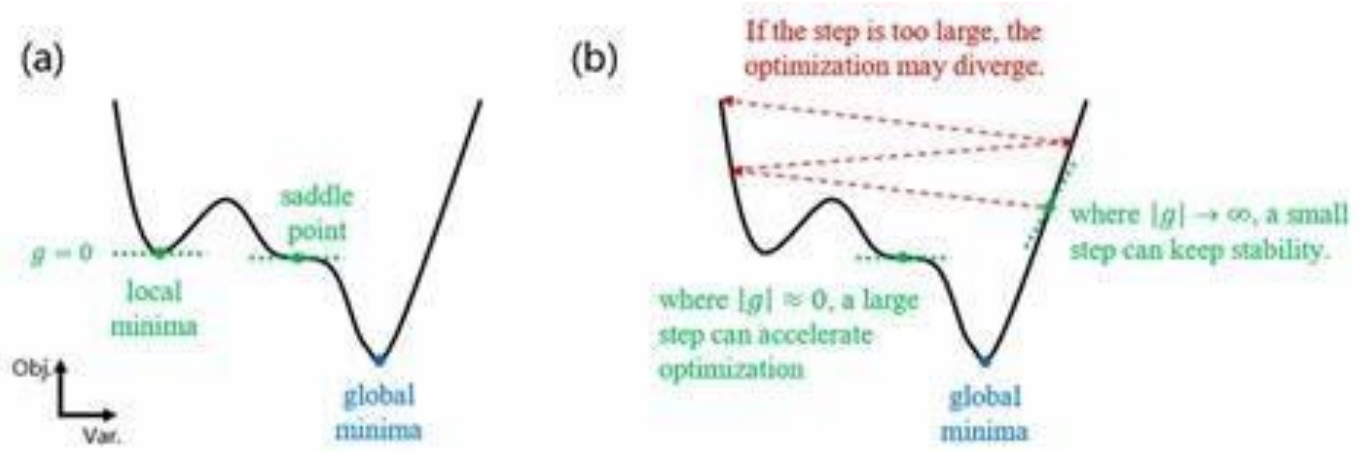


딥러닝은 그래서 batch 단위로 학습을 수행함
batch 단위에서 최소가 되게 하고 싶은 지표를
loss function 이라 함

학습이 한번에 될 수 없으니
loss function의 기울기를 가지고
모델의 weight를 수정하는 형태로 학습을 함

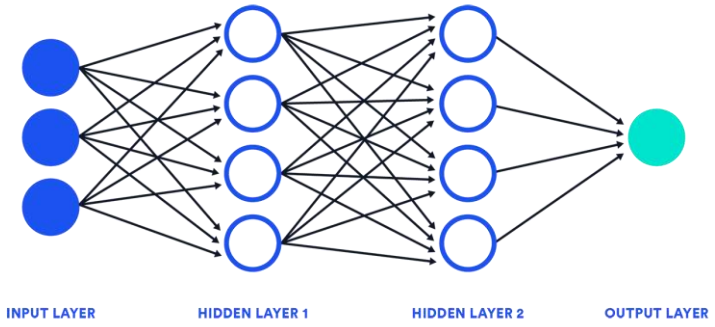
이게 경사하강법





손실함수를 통해서 기울기를 계산하고
weight를 갱신하는데
이 때 갱신하는 정도를 학습율(learning rate)이라고 함

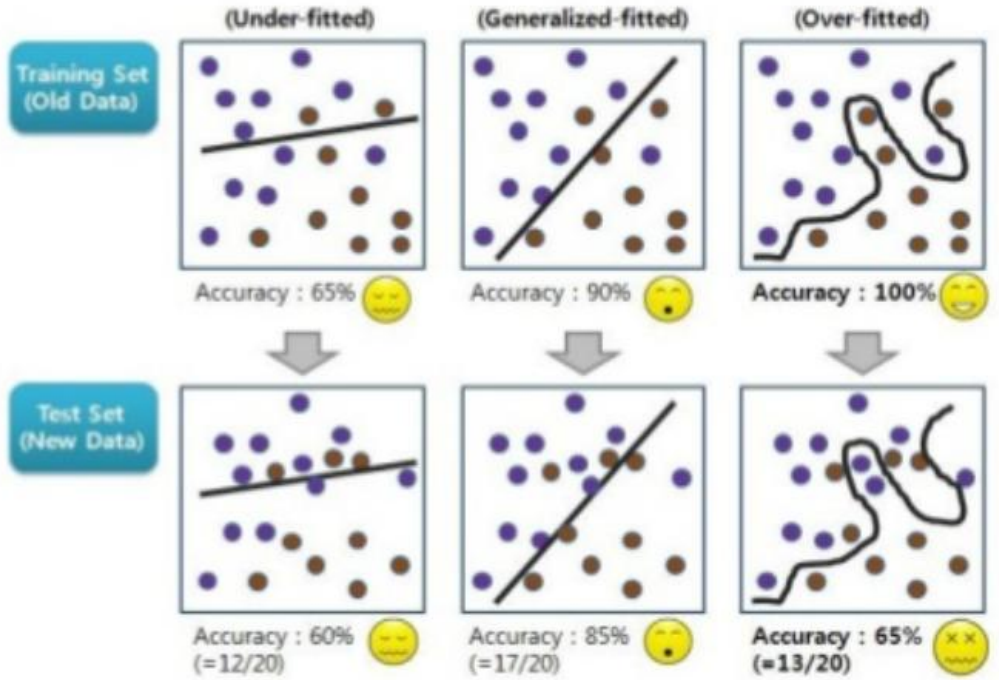
학습율이 높으면 global minima 에 안가고 진동하고,
낮으면 local minima를 못벗어남



딥러닝 학습은 그래서
batch 크기의 데이터에서 output 나오면
loss function 으로 실제값과 계산하고
기울기를 구한다음
output layer부터 weight를 업데이트하는 과정을
무수히 반복해서
Global minima 에 도달하게 하는 과정임

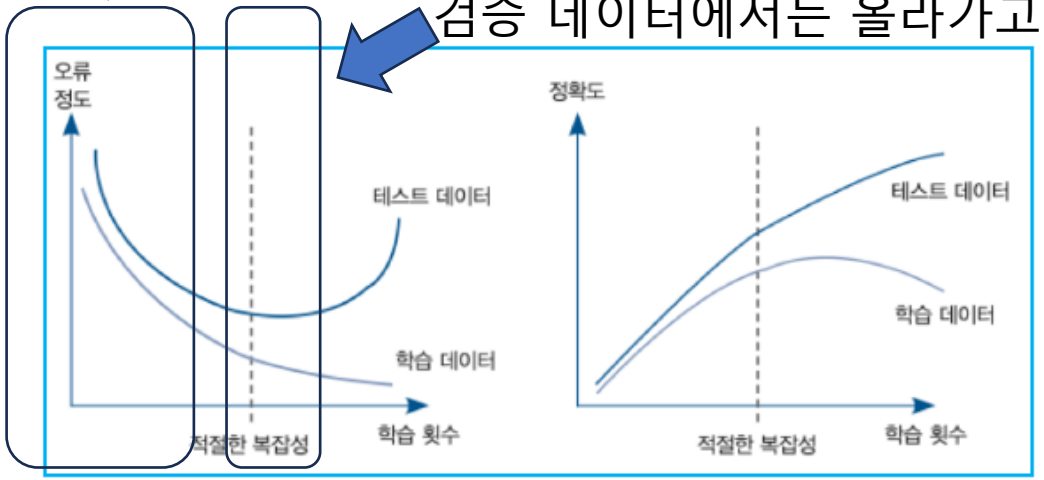
과소적합
- 말그대로 학습이 덜 된 상태
- Train성능도 덜나오고 Test 성능도 덜나옴

과적합
- 모델이 Train데이터에 대해 너무 과하게 학습된 상태



여기까지만 학습했다면
누가봐도 아직 성능 향상의 여지가 있어보임
이런 상태가 과소적합

학습데이터 Loss 는 떨어지는데
검증 데이터에서는 올라가고 있음



위 테스트 데이터 기재된 것은 validation Data임

Train 데이터와 Test 데이터가 완전히 패턴이 동일하다면
과적합해도 Test 데이터를 잘 맞추겠지만

현실적으로 다름

그래서 적당히 학습을 해야 됨

과적합 방지 설명

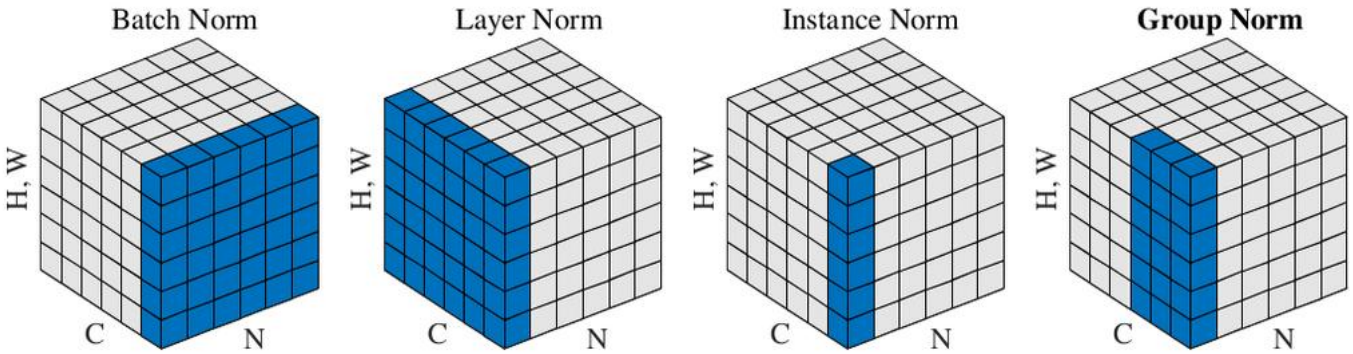
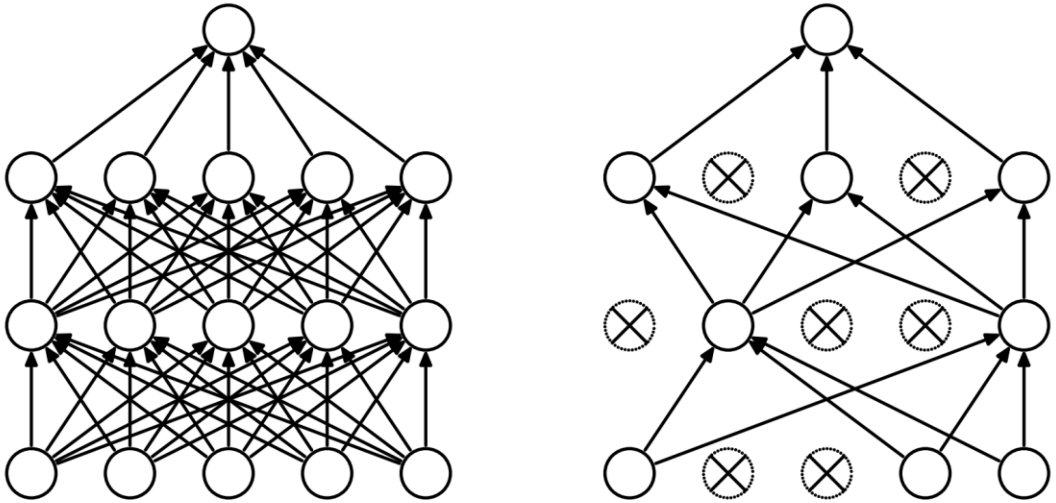


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

정규화

특정 기준으로 데이터를 정규화 하는 기법

Batch 는 Batch와 채널 크기 기준으로 정규화를 수행
Layer Norm 은 채널과 너비 높이 기준으로 정규화를 수행
Instance Norm 은 채널별로 정규화 수행



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

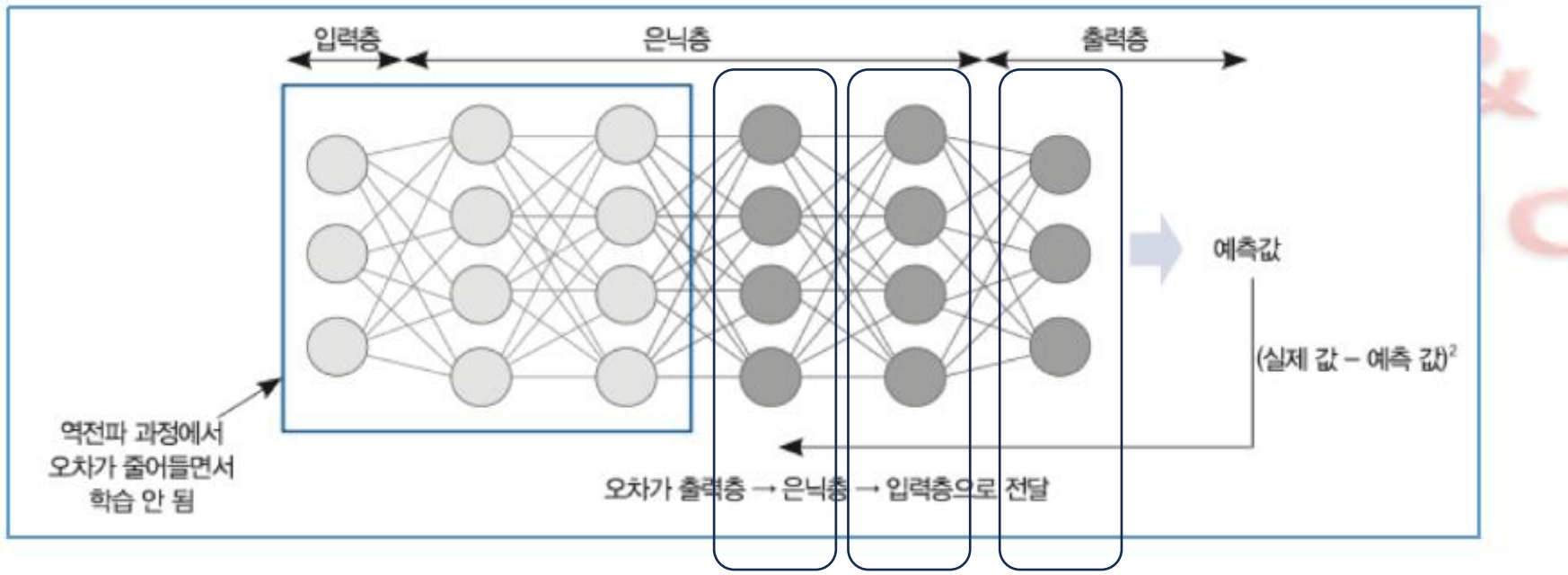
드롭아웃 아이디어

특정 노드가 최종 분류에 너무 영향을 많이 줌
> 그 노드가 틀리면 틀릴 확률이 너무 높아짐
> 그러면 학습 과정에서 랜덤하게 노드를 비활성화 시켜서
> 모든 노드가 최종 분류에 기여하게 하자!

기울기 소실 문제

레이어 깊게 쌓으면 성능 계속 좋아질 줄 알았는데...

오차 역전파 과정에서 변화율이 0에 가까워져서 가중치가 업데이트가 안되네..



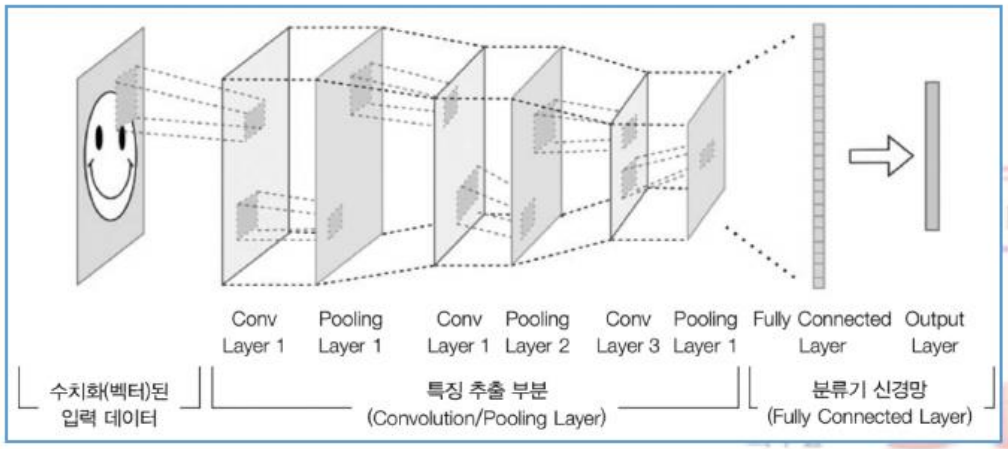
이걸 개선한 게
Res Net

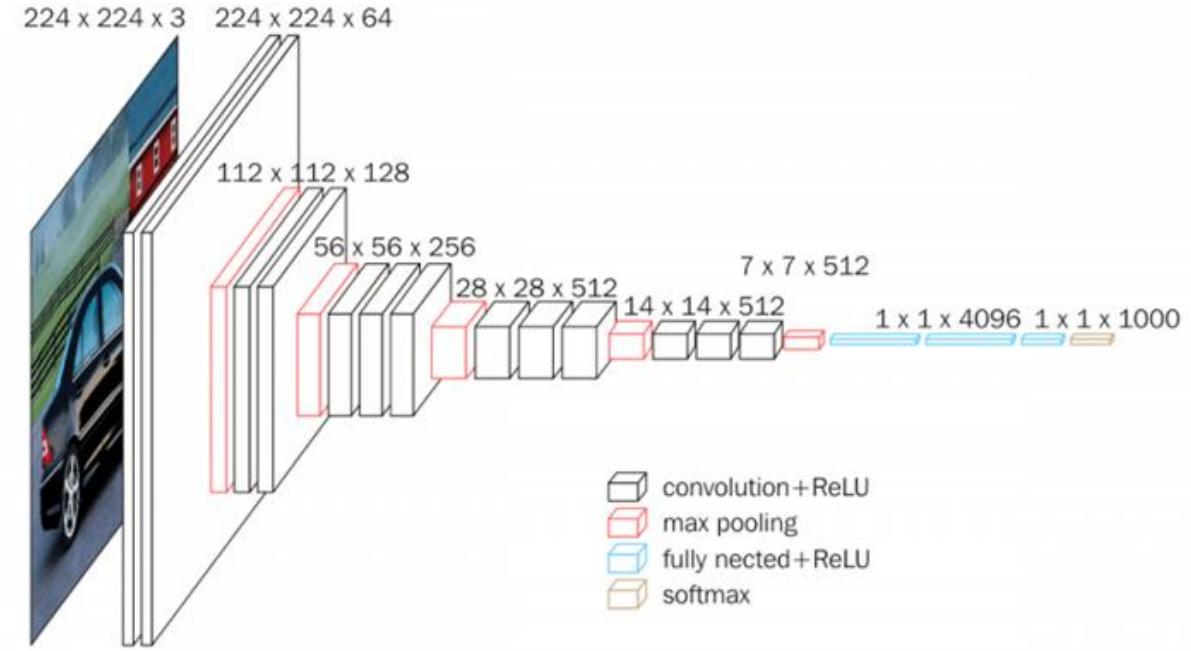
거의 업데이트 안됨

학습이 안됨..
성능 개선이 없음

모델 weight 가
어느정도 업데이트 됨

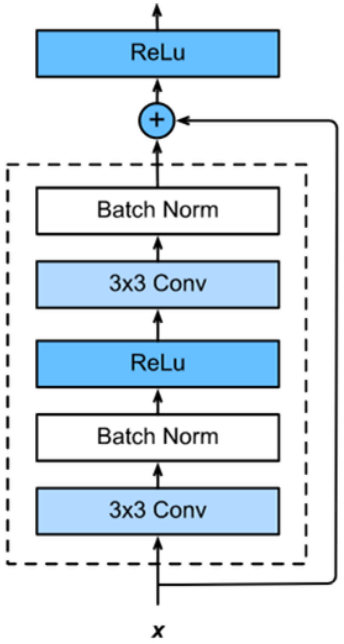
컴퓨터 비전 모델

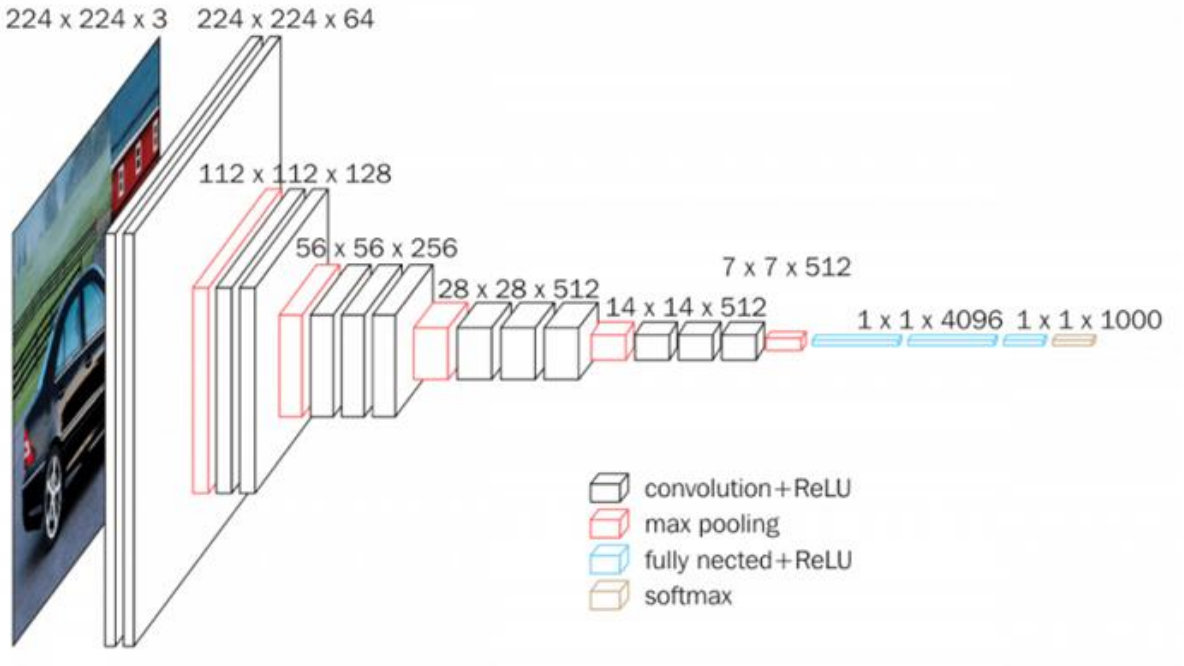




레이어도 중복이 되니까
중복되는 부분을 묶어서 블록이라 부르게 됨

예시)
Convolution + Normalization+ Activation + pooling
한세트로 묶어서 블록으로 부름





이미지 분류 모델은
그래서 Convolution + normalization + Activation + Pool 순으로
여러 층 쌓다가
Flatten 해서 이미지 개수로 차원을 줄인 다음 Softmax 활성화 함수를 통해 하는 형태로 함

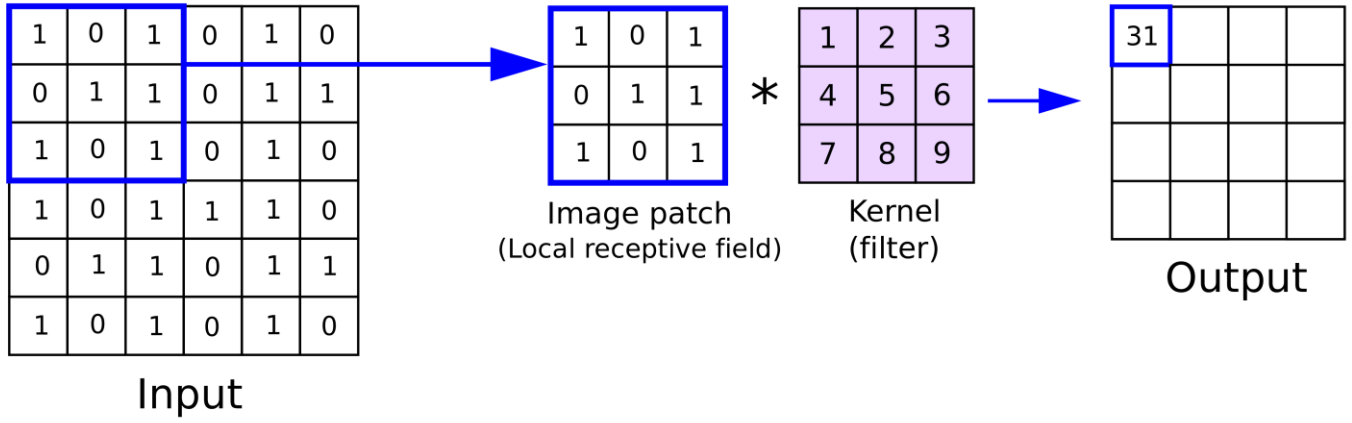
1.정의

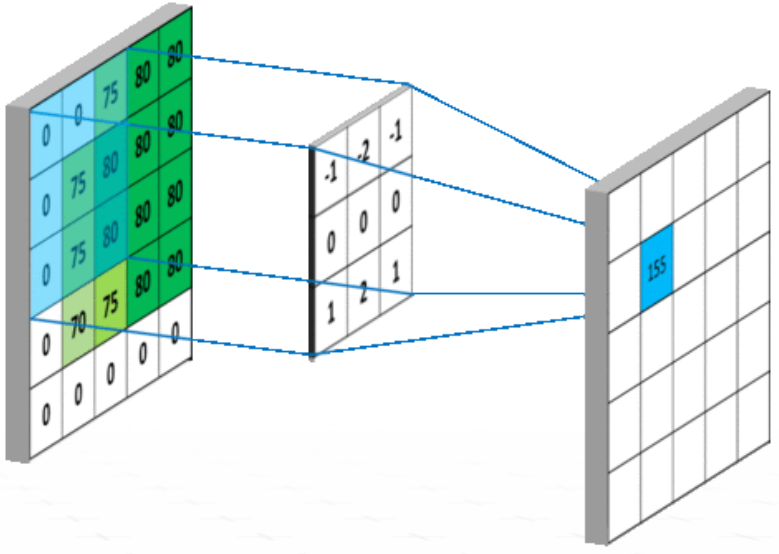
1) 합성곱 층(Convolutional layer)

① 필터(kernel)을 사용하여 입력 데이터를 슬라이딩하면서 합성곱 연산을 수행하는 레이어

② 합성곱 연산: 커널값과 커널크기의 입력값을 곱한 후 더하는 연산

- kernel size: 풀링에서 이동하는 윈도우 크기
- stride: 윈도우가 움직일 때 이동하는 간격
- padding: 입력 데이터의 경계에 추가적인 영역을 생성해주는 방법
- dilation: 커널 간의 간격을 의미

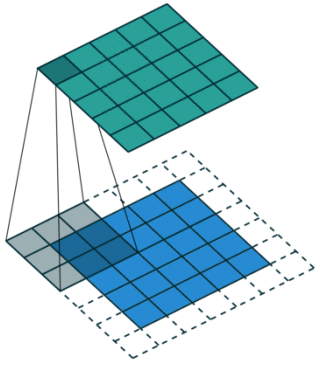




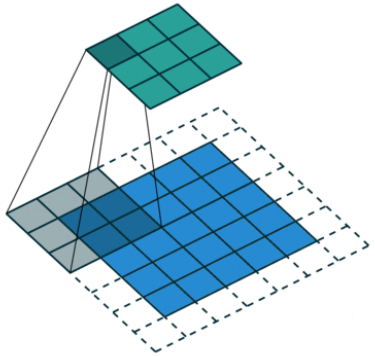
1.정의

1) 합성곱 층(Convolutional layer)

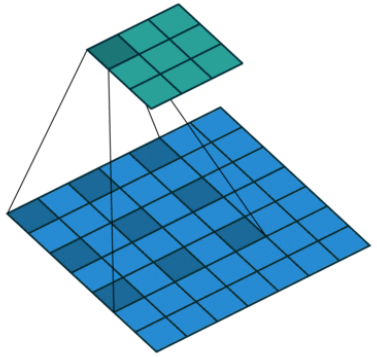
- kernel size: 풀링에서 이동하는 윈도우 크기
- stride: 윈도우가 움직일 때 이동하는 간격
- padding: 입력 데이터의 경계에 추가적인 영역을 생성해주는 방법
- dilation: 커널 간의 간격을 의미



kernel=3
stride =1
padding=1
dilated=1



kernel=3
stride =2
padding=1
dilated=1

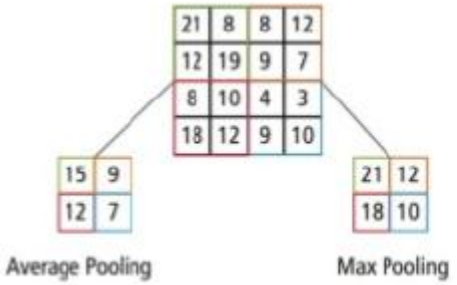


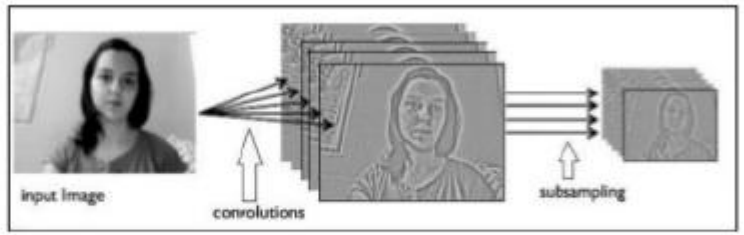
kernel=3
stride=1
padding=0
dilated =2

- **최대 풀링(Max Pooling)**: 윈도우 내 **최대값**을 선택 (가장 두드러진 특징 유지)
- **평균 풀링(Average Pooling)**: 윈도우 내 **평균값**을 선택 (전체적인 정보 유지)
- 확률 풀링 (Stochastic Pooling) : **값의 크기에 따라 확률적으로 선택**(작은 특징도 반영 가능, 계산량 증가)

역할

- **특징맵 크기 감소(차원 축소)** → 연산량 감소, 과적합 방지
- **위치 불변성 유지** → 이미지의 위치 변화에도 강건함

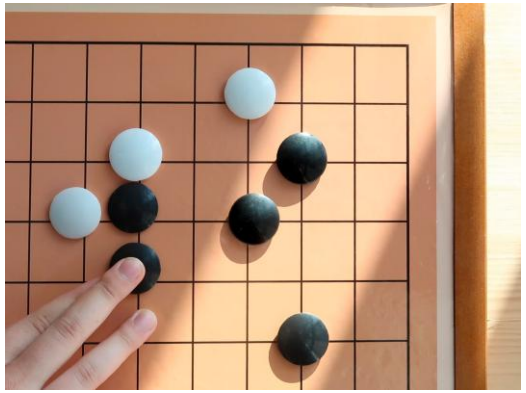




[Convolution후 Subsampling 처리]

평균 풀링 하면 좋을 줄 알고 사람들이 처음에 많이 테스트했었는데

이미지가 블러 처리된거 처럼 뭉게져서 오히려 성능이 안좋아짐
Max 로 연산하면 Max 값이 남아서 오히려 샤프하게 남음



강화학습 대충 설명하면

수백판 바둑을 두면서 이겼으면 베니핏을 주고 지면 페널티를 주게 함

그 과정에서 매 순간 상태(State)에 따라서 행동(Action)한 것에 대해 가치함수를 통해 평가