빅데이터 처리

-Big Data Processing

9주차

진로 특강

- **10월 31일 , 16:15 ~ (1시간 or 1시간 30분)**
- **강사 : ㈜ 포스로직 , 송종현 대표**
- **주제 : 성공 적인 커리어를 쌓기 위한 준비**
  - 사회생활 전반에 대한 마인드셋
  - 미래를 풍요롭게 하기 위한 준비.
  - 일을 하는 이유
  - 어떤 회사를 선택 해야 할까?
- **강사 이력**
  - 2008 ~ 2010 주식회사 이노와이어리스 연구원
  - 2011 ~ 2019 주식회사 엠브이텍 연구소장
  - 2019 ~ 현재 주식회사 포스로직 대표이사

빅데이터 처리

- **평가**
  - 중간 평가 : 서술형 시험, 30%,
  - 과제 : 25%,
  - 기말 평가 : 데이터 분석 프로젝트 (개인), 35% , github 포트폴리오
- **계획**
  - 9주차 : 머신 러닝 분석
  - 10주차 : 데이터 시각화
  - 11주차 : 데이터 시각화
  - 12주차 : 지리 정보 분석
  - 13주차 : 텍스트 분석
  - 14주차 : 시계열 데이터 분석
  - 15주차

오늘 수업 순서

- 머신러닝을 이용한 데이터 분석 process
- 하이퍼 파라미터 튜닝
- 부스팅 알고리즘
  - ada-boost, xgboost, lgbm
- 선형 회귀, SVM
- 피마 인디안 당뇨병 예측 : 분류 실습
- 보스턴 집값 : 회귀 예측
- 보팅, feature importance
- 시험지 확인

과제

분류: iris data
회귀 : 2차원 노이즈 데이터

알고리즘 : 의사 결정 트리 (decision tree)
랜덤 포레스트 (Random Forest)

Code: dc_rf_final_nofill.ipynb

1) 의사 결정 트리의 분류와 회귀 code에서 max_depth가 3일때의 tree image 를 제출 하시오 (ppt에 copy)

2) 아래의 성능 table을 완성 하시오

분류 Iris data	Decision tree	Decision tree	RandomForest	RandomForest
	max_depth=2	max_depth=3	n_estimators=4	n_estimators=30
accuracy	0.93	0.96	0.953	0.966

회귀 2d noise data	Decision tree	Decision tree	RandomForest	RandomForest
	max_depth=2	max_depth=3	n_estimators=5	n_estimators=50
RMSE	0.172	0.127	0.121	0.1161

머신 러닝을 이용한 데이터 분석 process

판매량 예측

학습 데이터

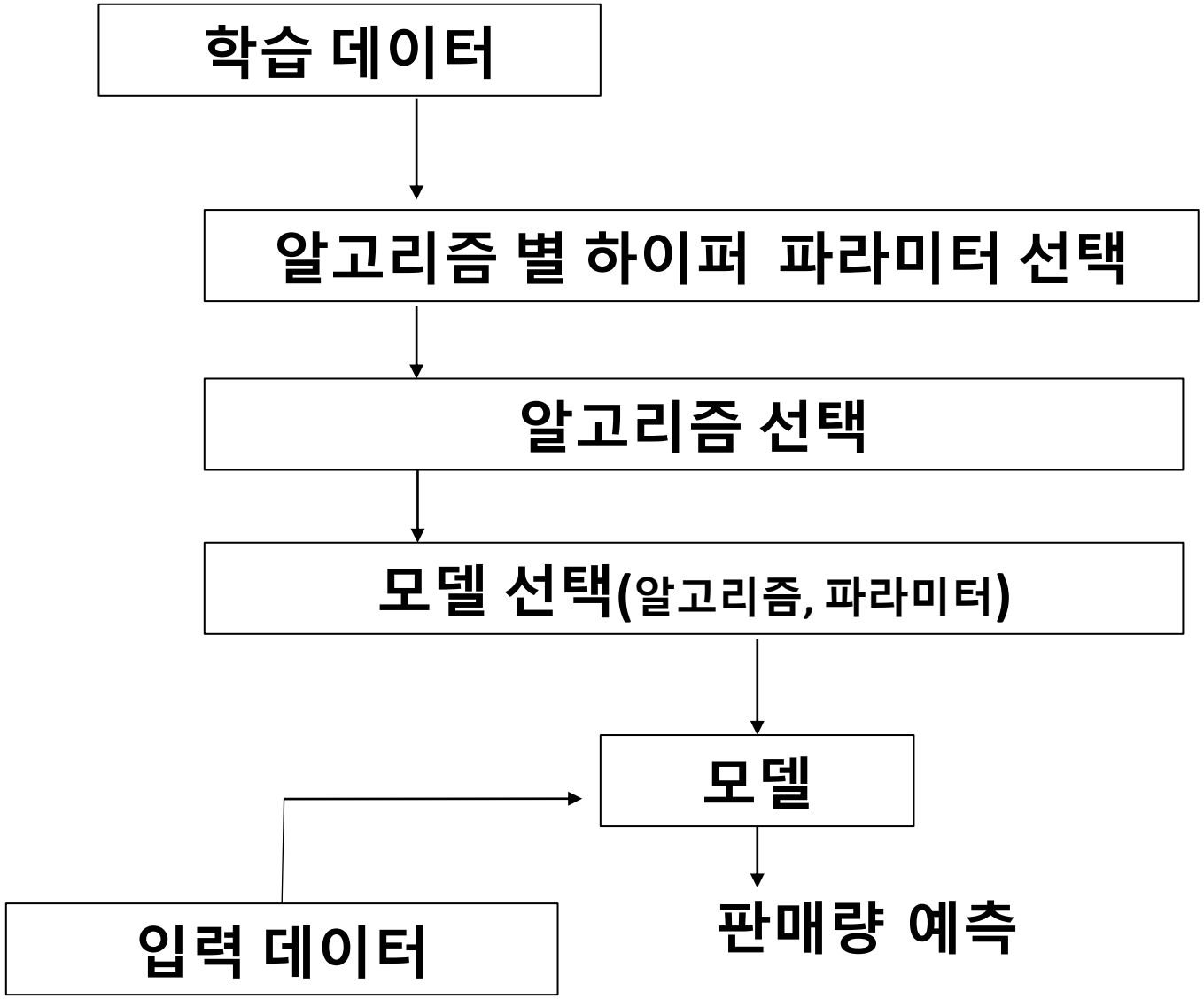
날짜	요일	온도	판매량

입력
독립변수
특성
속성
Feature

출력 (target)
종속 변수

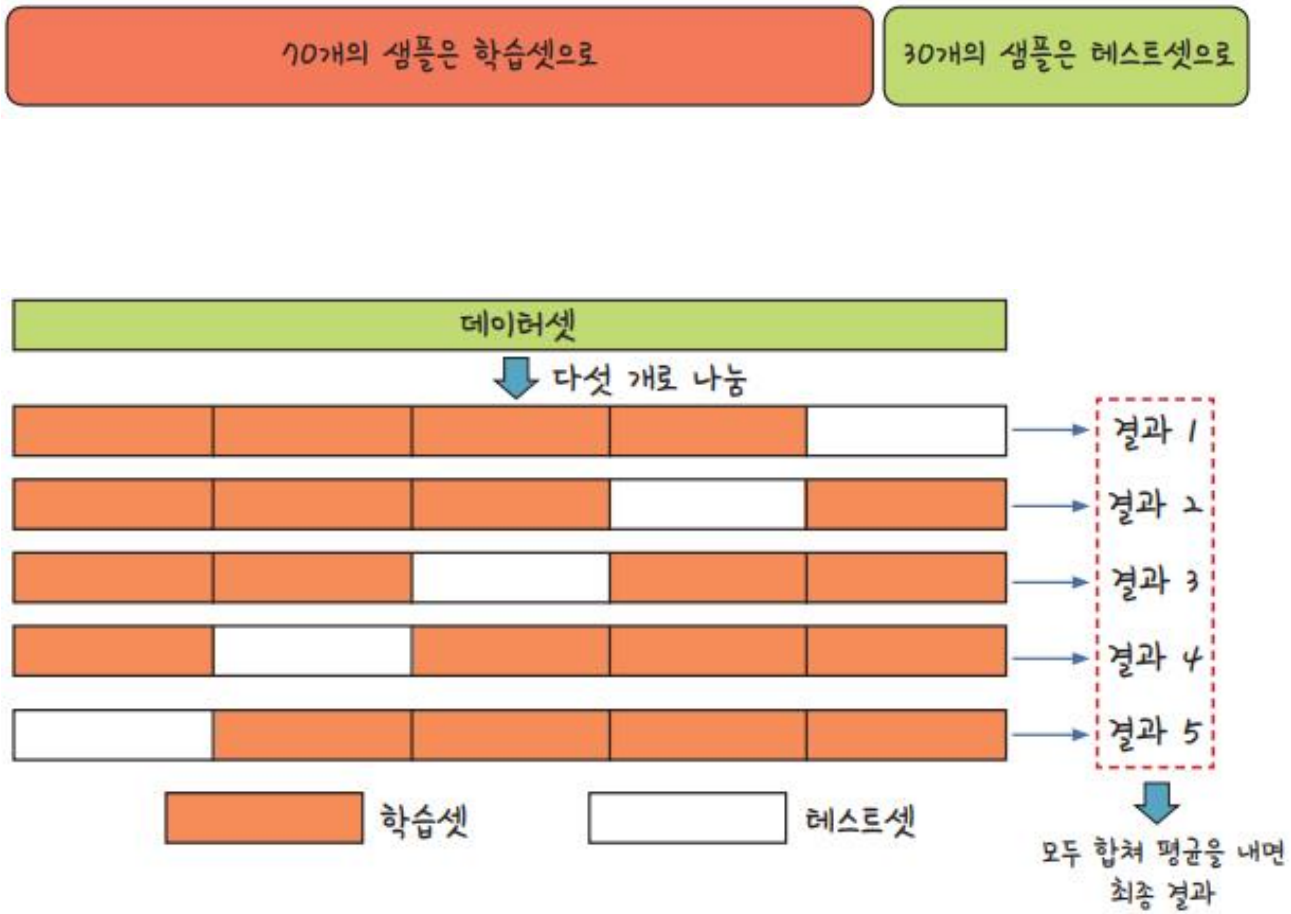
입력 데이터

날짜	요일	온도



성능 비교

교차검증


70개의 샘플은 학습셋으로
30개의 샘플은 테스트셋으로
데이터셋
다섯 개로 나눔
결과 1
결과 2
결과 3
결과 4
결과 5
학습셋
테스트셋
모두 합쳐 평균을 내면
최종 결과

그리드 서치 (Grid Search)

하이퍼 파라미터 탐색 (교차 검증 기반)

예) 랜덤 포레스트 알고리즘
: n_estimator: 100,200
min_samples_leaf : 3,5,7

아래 조합중 가장 성능이 좋은 조합을 찾아냄
(100,3),(100,5),(100,7),(200,3),(200,5),(200,7)

그리드 서치 (Grid Search)

- 하이퍼 파라미터 튜닝
  - 성능을 최적화 하는 parameter 선택 방법

- 의사 결정 트리의 grid search 예제

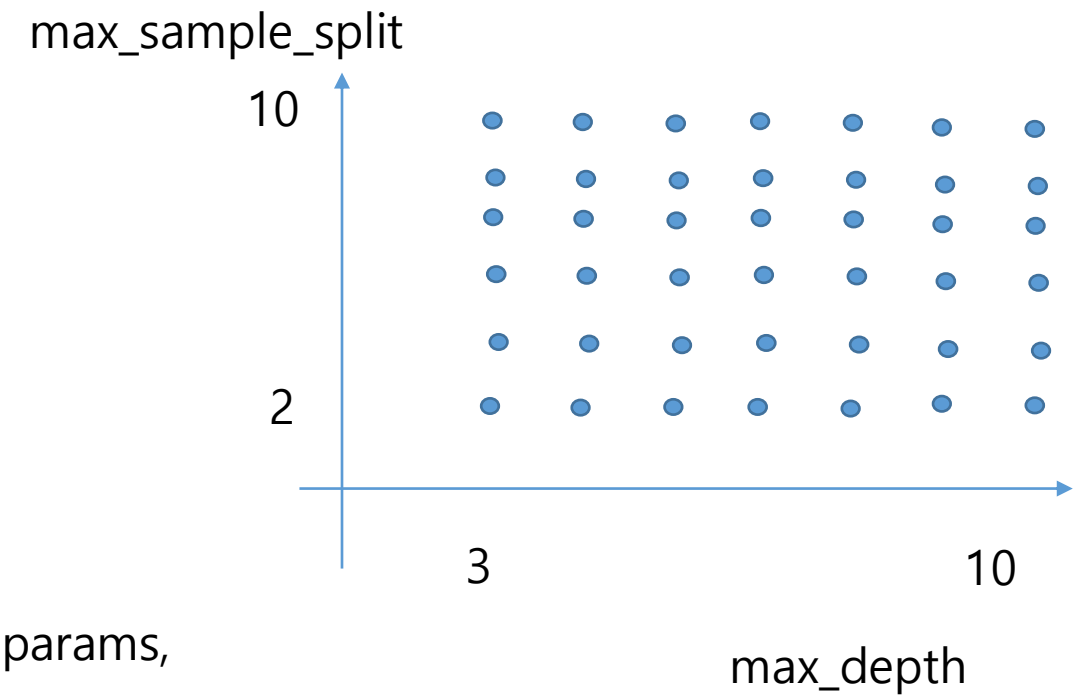
Parameter

max_depth : 결정 트리 최대 깊이
max_sample_split: node를 나누기 위한 최소 샘플 개수

그리드 서치 (Grid Search)


```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target
params = {'max_depth': range(3, 10, 1),
          'min_samples_split': range(2, 10, 1)
          }
gs = GridSearchCV(DecisionTreeClassifier(random_state=42), params,
n_jobs=-1)
gs.fit(X, y)
print(gs.best_params_)
print(gs.best_score_)
```

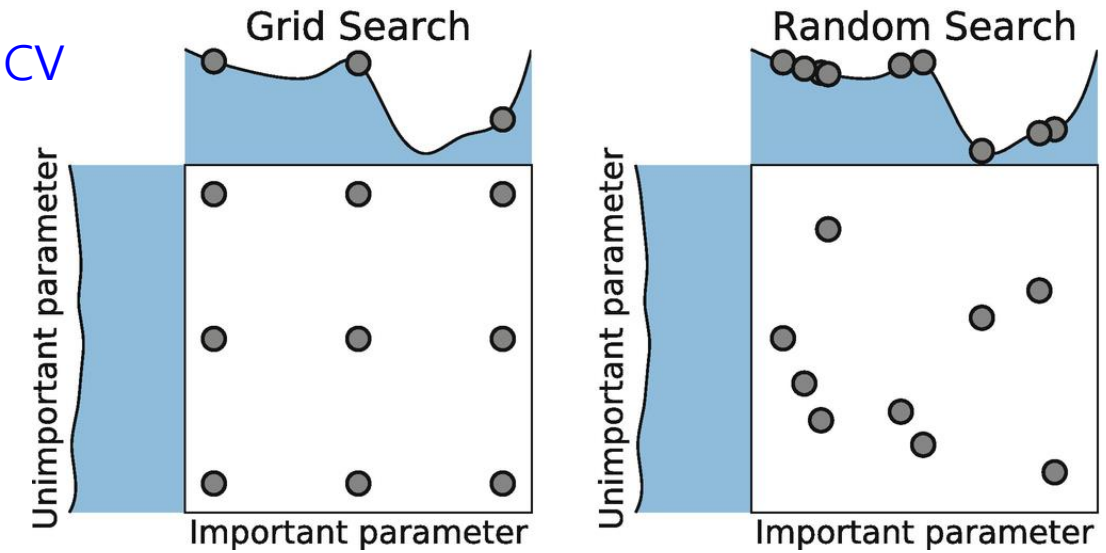



```
{'max_depth': 4, 'min_samples_split': 2}
0.9666666666666668
```

랜덤 서치 (Random Search)


```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint


iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target
params = {'max_depth': randint(3, 10),
          'min_samples_split': randint(2, 10)
          }
gs = RandomizedSearchCV(DecisionTreeClassifier(random_state=42),
params,n_iter=10, n_jobs=-1)
gs.fit(X, y)
print(gs.best_params_)
print(gs.best_score_)
```



알고리즘 성능 비교

- **분류 (Classification)**
  - 알고리즘: adaboost, Xgboost, LGBM, SVM, (Support Vector Machine)
  - 데이터: 당뇨병 데이터
- **회귀 (Regression)**
  - 에이다부스트, XgBoost, LGBM, SVM (Support Vector Machine) ,선형 회귀
  - 데이터 : 보스턴 집값 예측

당뇨병 데이터 - 분류

- **피마 인디언 데이터**

속성 / 클래스

	정보 1	정보 2	정보 3	…	정보 8	당뇨병 여부
1번째 인디언	6	148	72	…	50	1
2번째 인디언	1	85	66	…	31	0
3번째 인디언	8	183	64	…	32	1
…	…	…	…	…	…	…
768번째 인디언	1	93	70	…	23	0

샘플

- 샘플 수: 768
- 속성: 8
  - 정보 1(pregnant): 과거 임신 횟수
  - 정보 2(plasma): 포도당 부하 검사 2시간 후 공복 혈당 농도(mm Hg)
  - 정보 3(pressure): 확장기 혈압(mm Hg)
  - 정보 4(thickness): 삼두근 피부 주름 두께(mm)
  - 정보 5(insulin): 혈청 인슐린(2-hour, mu U/ml)
  - 정보 6(BMI): 체질량 지수(BMI, weight in kg/(height in m)^2)
  - 정보 7(pedigree): 당뇨병 가족력
  - 정보 8(age): 나이
- 클래스: 당뇨(1), 당뇨 아님(0)

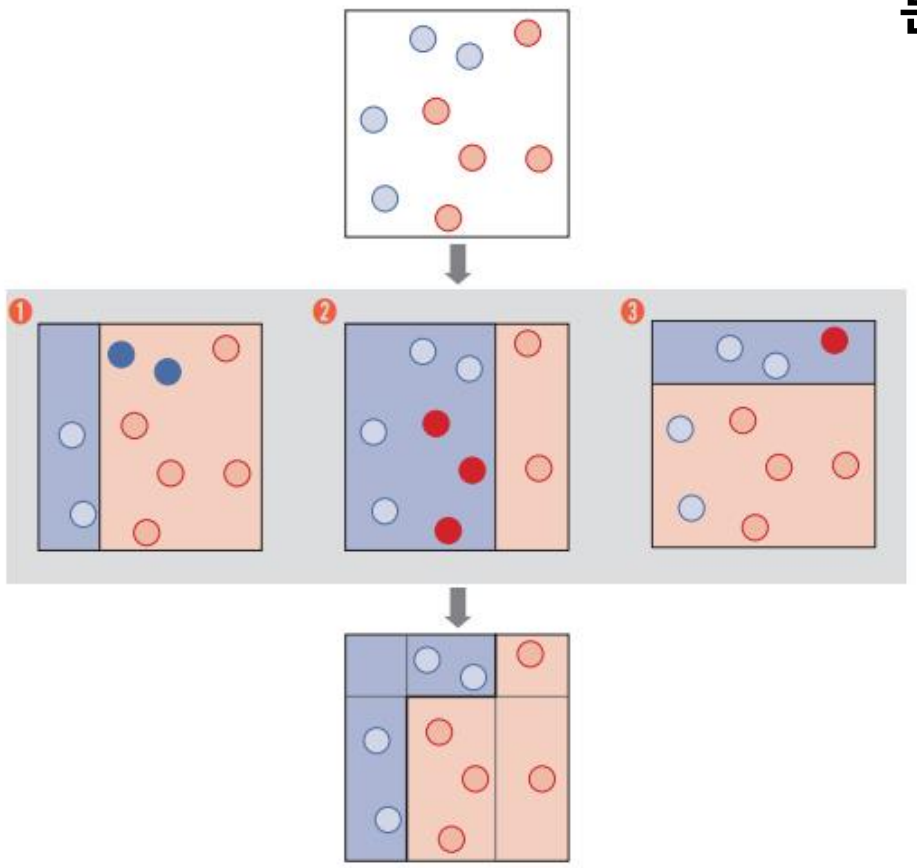
참고 모두의 딥러닝

보스턴 집값 예측- 회귀

▪ 보스턴 시 주택 가격 정보

[01] CRIM	자치시(town) 별 1인당 범죄율
[02] ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03] INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04] CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05] NOX	10ppm 당 농축 일산화질소
[06] RM	주택 1가구당 평균 방의 개수
[07] AGE	1940년 이전에 건축된 소유주택의 비율
[08] DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09] RAD	방사형 도로까지의 접근성 지수
[10] TAX	10,000 달러 당 재산세율
[11] PTRATIO	자치시(town)별 학생/교사 비율
[12] B	1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13] LSTAT	모집단의 하위계층의 비율(%)
[14] MEDV	본인 소유의 주택가격(중앙값) (단위: $1,000)

에이다 부스트 (Ada boost)



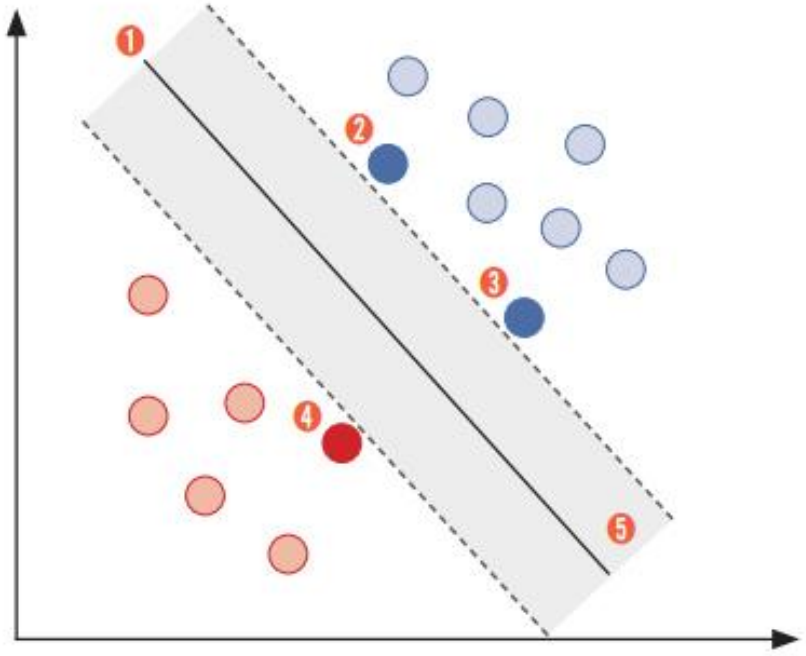
분류와 회귀에 사용

- 에이다 부스트(ada boost)는 여러 번의 분류를 통해 정답을 예측해 가는 알고리즘
- 여러 개의 분류기를 합쳐서 성능 높은 분류기를 생성
- 예를 들어 1)처럼 분류해 보니 상단의 푸른 원 두 개가 잘못 예측된 것을 알 수 있음
- 잘못 예측된 이 두 개에 가중치를 두고 2)처럼 다시 분류
- 하단의 붉은 원 세 개가 잘못되어 있음
- 이 세 개에 다시 가중치를 두고 3)처럼 분류
- 이와 같은 반복 훈련 후 모든 분류(1) ,2) 3)) 결과를 합산하면 푸른 원과 붉은 원을 구분하는 분류기가 완성

sklearn.ensemble	**AdaBoostRegressor**	**AdaBoostClassifier**

참고 모두의 딥러닝

서포트 벡터 머신 (Support Vector Machine)



- 분류와 회귀에 사용
- 서포트 벡터 머신(support vector machine)은 분류를 위한 기준선을 정의하는 모델
- 다음 그림에서 빨간 점과 파란 점을 구분하는 경계선(1))을 만든다고 할 때, 경계선은 파란색 두 점(2) , 3))과 빨간색 점(4))에서 가장 떨어진 곳에 위치해야 할 것
- 이 세 점(2) , 3) , 4))을 서포트 벡터라고 하며, 이로 인해 만들어지는 공간(5))을 마진이라고 함
- 마진이 최대화되는 경계를 찾아 분류를 하는 방법

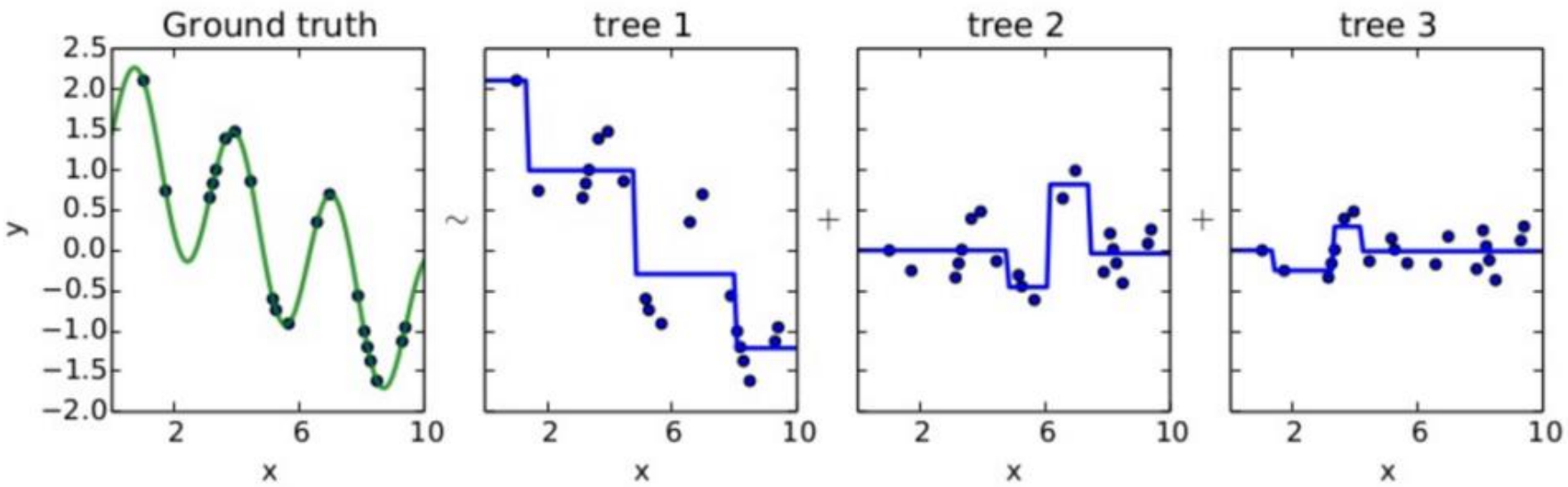
참고 모두의 딥러닝

sklearn.svm import SVC, SVR

동영상

XGBoost

- 그레디언트 부스팅
  - 깊이가 얕은 결정 트리를 사용하여 이전 트리의 오차를 보완
  - 경사 하강법 사용
- 그레디언트 부스팅 알고리즘을 최적화
- 분류과 회귀에 사용

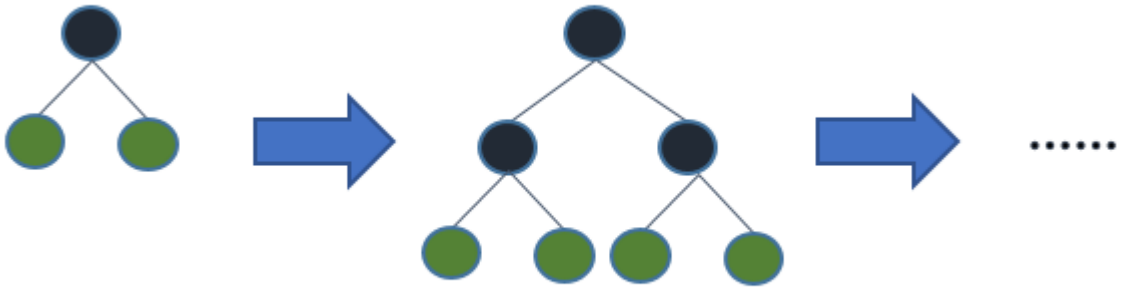


Library : xgboost XGBRegressor , XGBClassifier

LIGHT GBM(LGBM)

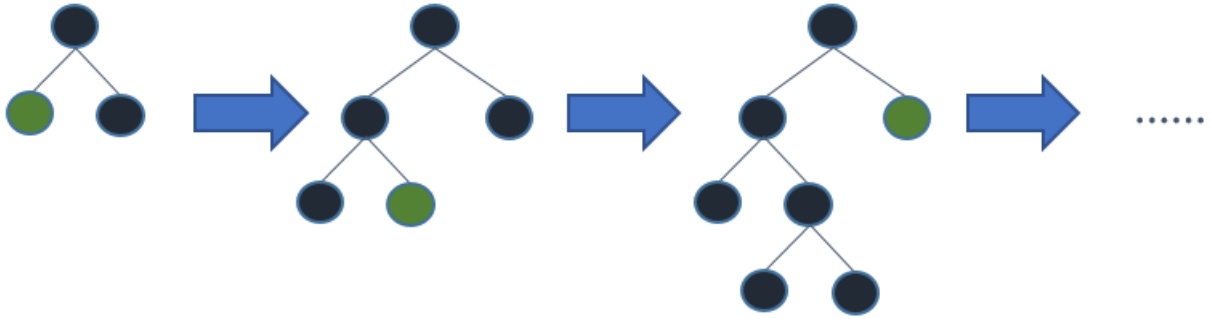
- **XGBoost의 학습 시간을 단축 시킨 알고리즘**

xgboost



Level-wise tree growth

Light gbm

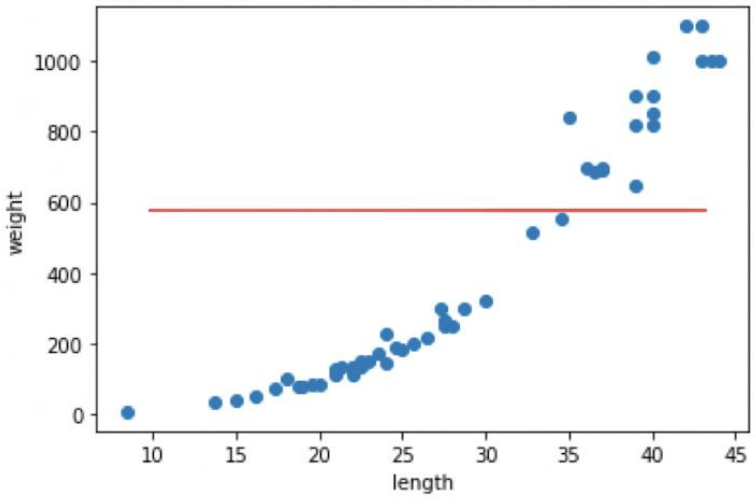


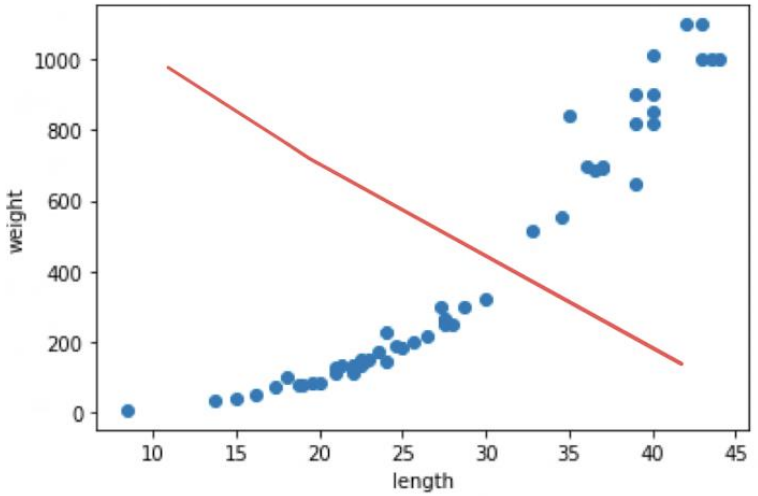
Leaf-wise tree growth

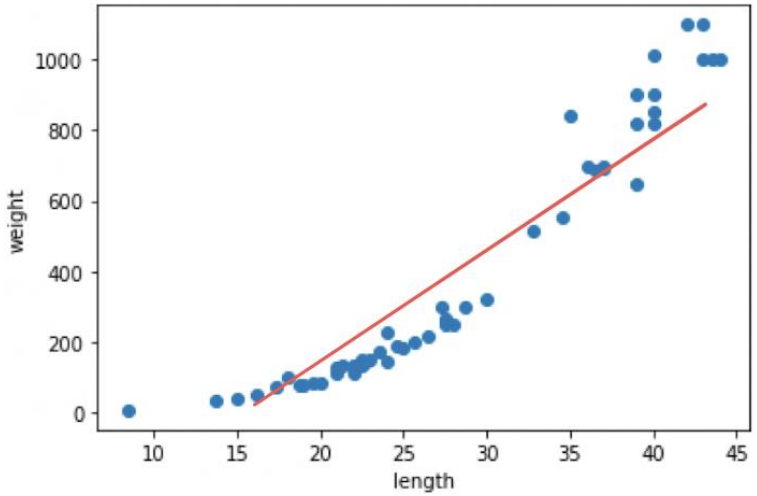
Library : **lightgbm** , **LGBMRegressor, LGBMClassifier**

선형 회귀 (Linear Regression)

- 특성을 잘 나타내는 직선을 학습 하는 알고리즘
- 회귀에만 사용







sklearn. linear_model **LinearRegression**

참고 핸즈온 머신러닝

당뇨병 데이터 –adaboost


```
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier


# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('/content/pima-indians-diabetes3.csv')

# 세부 정보를 X로 지정합니다.
X = df.iloc[:,0:8]
# 당뇨병 여부를 Y로 지정합니다.
y = df.iloc[:,8]

al = AdaBoostClassifier()
al.fit(X, y)

cscore=cross_val_score(al,X,y,cv=5)  # 교차 검증 k=5
print('accuracy',cscore.mean())
```

base_estimator :
Default : decision tree

n_estimators

당뇨병 데이터 –support vector machine


```
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split,
cross_val_score, StratifiedKFold

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('/content/pima-indians-diabetes3.csv')

# 세부 정보를 X로 지정합니다.
X = df.iloc[:,0:8]
# 당뇨병 여부를 Y로 지정합니다.
y = df.iloc[:,8]

al = SVC()
al.fit(X, y)

cscore=cross_val_score(al,X,y,cv=5)  # 교차 검증 k=5
print('accuracy',cscore.mean())
X.shape
```

kernel{‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’} or call

당뇨병 데이터 –XGboost


```
import pandas as pd
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split,
cross_val_score, StratifiedKFold


# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('/content/pima-indians-diabetes3.csv')

# 세부 정보를 X로 지정합니다.
X = df.iloc[:,0:8]
# 당뇨병 여부를 Y로 지정합니다.
y = df.iloc[:,8]

al = XGBClassifier()
al.fit(X, y)

cscore=cross_val_score(al,X,y,cv=5)  # 교차 검증 k=5
print('accuracy',cscore.mean())
X.shape
```

max_depth: default 8

당뇨병 데이터 –LGBM


```
import pandas as pd
from lightgbm import LGBMClassifier
from sklearn.model_selection import train_test_split,
cross_val_score, StratifiedKFold

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('/content/pima-indians-diabetes3.csv')

# 세부 정보를 X로 지정합니다.
X = df.iloc[:,0:8]
# 당뇨병 여부를 Y로 지정합니다.
y = df.iloc[:,8]

al = LGBMClassifier()
al.fit(X, y)

cscore=cross_val_score(al,X,y,cv=5)  # 교차 검증 k=5
print('accuracy',cscore.mean())
```

learning_rate : 0.01 ~ 0.1
num_iterations=default : 100

보스턴 집값 -SVR


```
from sklearn.model_selection import cross_val_score
import pandas as pd
from sklearn.svm import SVR
import pandas as pd
import numpy as np



raw_df = pd.read_csv('/content/Boston_house.csv')

y_target = raw_df['Target']
X_data = raw_df.drop(['Target'], axis=1,inplace=False)
```

```
rf = SVR()
neg_mse_scores = cross_val_score(rf, X_data, y_target,
scoring="neg_mean_squared_error", cv = 5)
rmse_scores  = np.sqrt(-1 * neg_mse_scores)
avg_rmse = np.mean(rmse_scores)

print(' 5 교차 검증의 개별 Negative MSE scores: ',
np.round(neg_mse_scores, 2))
print(' 5 교차 검증의 개별 RMSE scores : ',
np.round(rmse_scores, 2))
print(' 5 교차 검증의 평균 RMSE : {0:.3f}
'.format(avg_rmse))
```

보스턴 집값 : 알고리즘 성능 비교


```
def get_model_cv_prediction(model, X_data, y_target):
    neg_mse_scores = cross_val_score(model, X_data, y_target,
scoring="neg_mean_squared_error", cv = 5)
    rmse_scores  = np.sqrt(-1 * neg_mse_scores)
    avg_rmse = np.mean(rmse_scores)
    print('##### ',model.__class__.__name__ , ' #####')
    print(' 5 교차 검증의 평균 RMSE : {0:.3f} '.format(avg_rmse))
```

보스턴 집값 : 알고리즘 성능 비교


```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import AdaBoostRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor

raw_df = pd.read_csv('/content/Boston_house.csv')

y_target = raw_df['Target']
X_data = raw_df.drop(['Target'], axis=1,inplace=False)

lr_reg = LinearRegression()
svr_reg = SVR()
ada_reg = AdaBoostRegressor()
xgb_reg = XGBRegressor()
lgb_reg = LGBMRegressor()

# 트리 기반의 회귀 모델을 반복하면서 평가 수행
models = [lr_reg, svr_reg,ada_reg,  xgb_reg, lgb_reg]
for model in models:
    get_model_cv_prediction(model, X_data, y_target)
```

그리드 서치 과제

Iris 데이터에 random forest 알고리즘으로 분류 할때
Grid search를 이용하여 아래 범위에서 제일 성능이 좋은 parameter를 찾으시오.

: n_estimator: 4,5,6
  min_samples_leaf : 3,4,5

```
from sklearn.ensemble import RandomForestClassifier

params = {'n_estimators': range(4,7,1),
          'min_samples_split': range(3, 6, 1)
          }
```

과제 알고리즘 성능 비교

인디안 당뇨병 예측

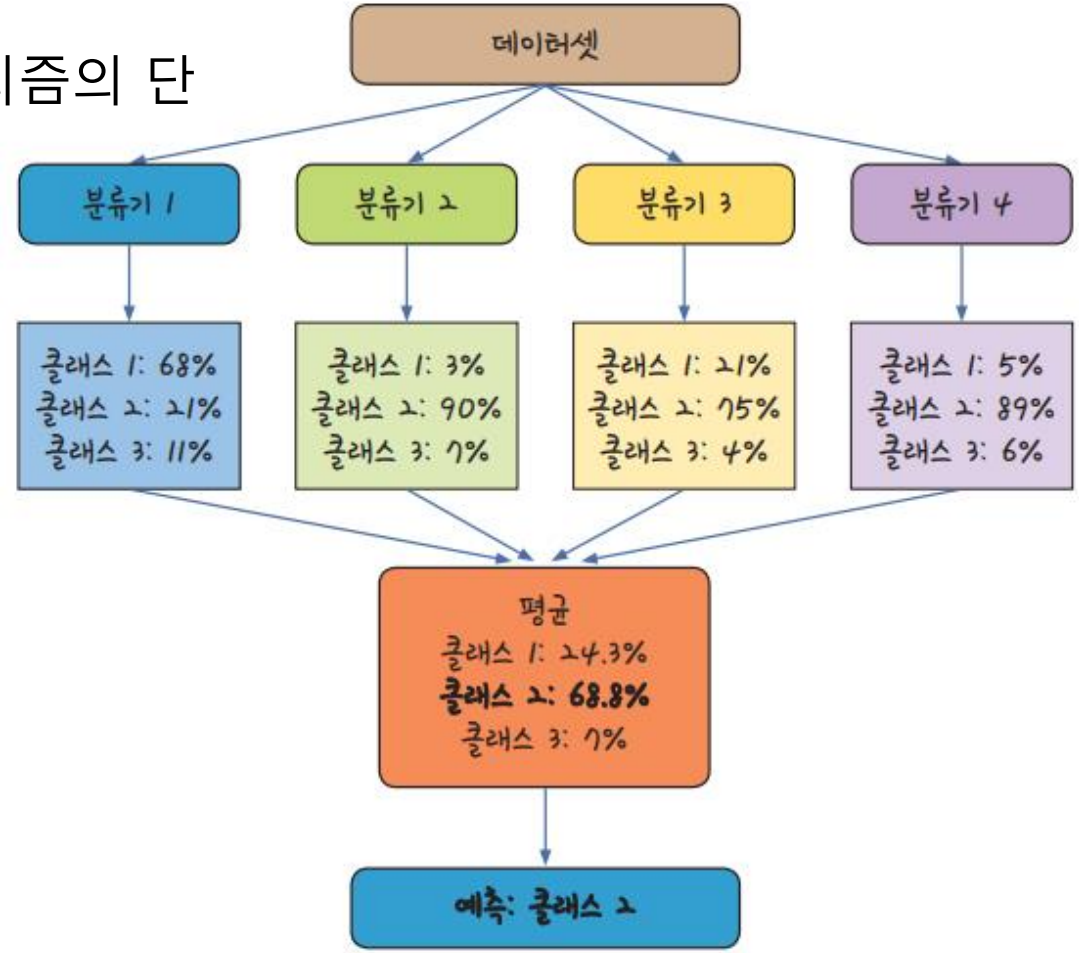
accuracy	adaboost	SVC	XGboost	LGBM	voting
Default					
Parameter 최적화 (선택)					

보스턴 집값 예측

RMSE	adaboost	SVC	XGboost	LGBM	Linear regression
Default					
Parameter 최적화 (선택)					

보팅 (Voting)

- 보팅(voting)은 여러 가지 다른 유형의 알고리즘을 같은 데이터 셋에 적용해 학습하는 방법
- 학습한 결과를 모아 다수의 분류기가 결정한 결과를 선택하거나 클래스별 평균을 종합해 예측
- 다양한 알고리즘을 분류기로 활용함으로써 단일 알고리즘의 단점을 극복할 수 있음



참고 모두의 딥러닝

보팅 (Voting)


```
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from lightgbm import LGBMClassifier
from sklearn.ensemble import VotingClassifier

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('/content/pima-indians-diabetes3.csv')

# 세부 정보를 X로 지정합니다.
X = df.iloc[:,0:8]
# 당뇨병 여부를 Y로 지정합니다.
y = df.iloc[:,8]

clf1 = AdaBoostClassifier()
clf2= SVC()
clf3=LGBMClassifier()
```

```
classifier=VotingClassifier(estimators=[('ada',clf1),('svc',clf
2),('lgbm',clf3)],verbose=True)
classifier.fit(X,y)


al.fit(X, y)

cscore=cross_val_score(classifier,X,y,cv=5)  # 교차 검증
k=5
print('accuracy',cscore.mean())
```

보팅 (Voting)


```
[Voting] ..... (1 of 3) Processing ada, total=   0.2s
[Voting] ..... (2 of 3) Processing svc, total=   0.1s
[Voting] ..... (3 of 3) Processing lgbm, total=   0.2s
[Voting] ..... (1 of 3) Processing ada, total=   0.2s
[Voting] ..... (2 of 3) Processing svc, total=   0.0s
[Voting] ..... (3 of 3) Processing lgbm, total=   0.1s
[Voting] ..... (1 of 3) Processing ada, total=   0.2s
[Voting] ..... (2 of 3) Processing svc, total=   0.1s
[Voting] ..... (3 of 3) Processing lgbm, total=   0.1s
[Voting] ..... (1 of 3) Processing ada, total=   0.1s
[Voting] ..... (2 of 3) Processing svc, total=   0.0s
[Voting] ..... (3 of 3) Processing lgbm, total=   0.1s
[Voting] ..... (1 of 3) Processing ada, total=   0.1s
[Voting] ..... (2 of 3) Processing svc, total=   0.0s
[Voting] ..... (3 of 3) Processing lgbm, total=   0.1s
[Voting] ..... (1 of 3) Processing ada, total=   0.1s
[Voting] ..... (2 of 3) Processing svc, total=   0.0s
[Voting] ..... (3 of 3) Processing lgbm, total=   0.1s
accuracy 0.7604872251931075
```

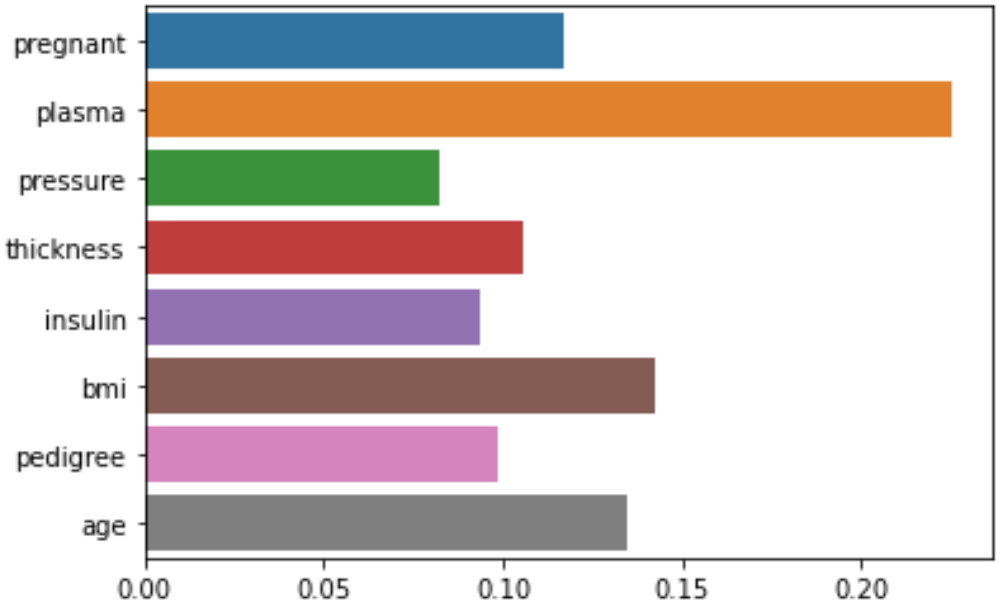
Feature importance

- **Feature (특성)의 중요도 표시**

```
xgb_cls = XGBClassifier(n_estimators=1000)

# 앞 예제에서 만들어진 X_data, y_target 데이터 셋을 적용하여 학습합니다.
xgb_cls.fit(X, y)

feature_series = pd.Series(data=xgb_cls.feature_importances_,
index=X.columns )
#feature_series = feature_series.sort_values(ascending=False)
sns.barplot(x= feature_series, y=feature_series.index)
```



머신 러닝을 이용한 데이터 분석 process

판매량 예측

학습 데이터

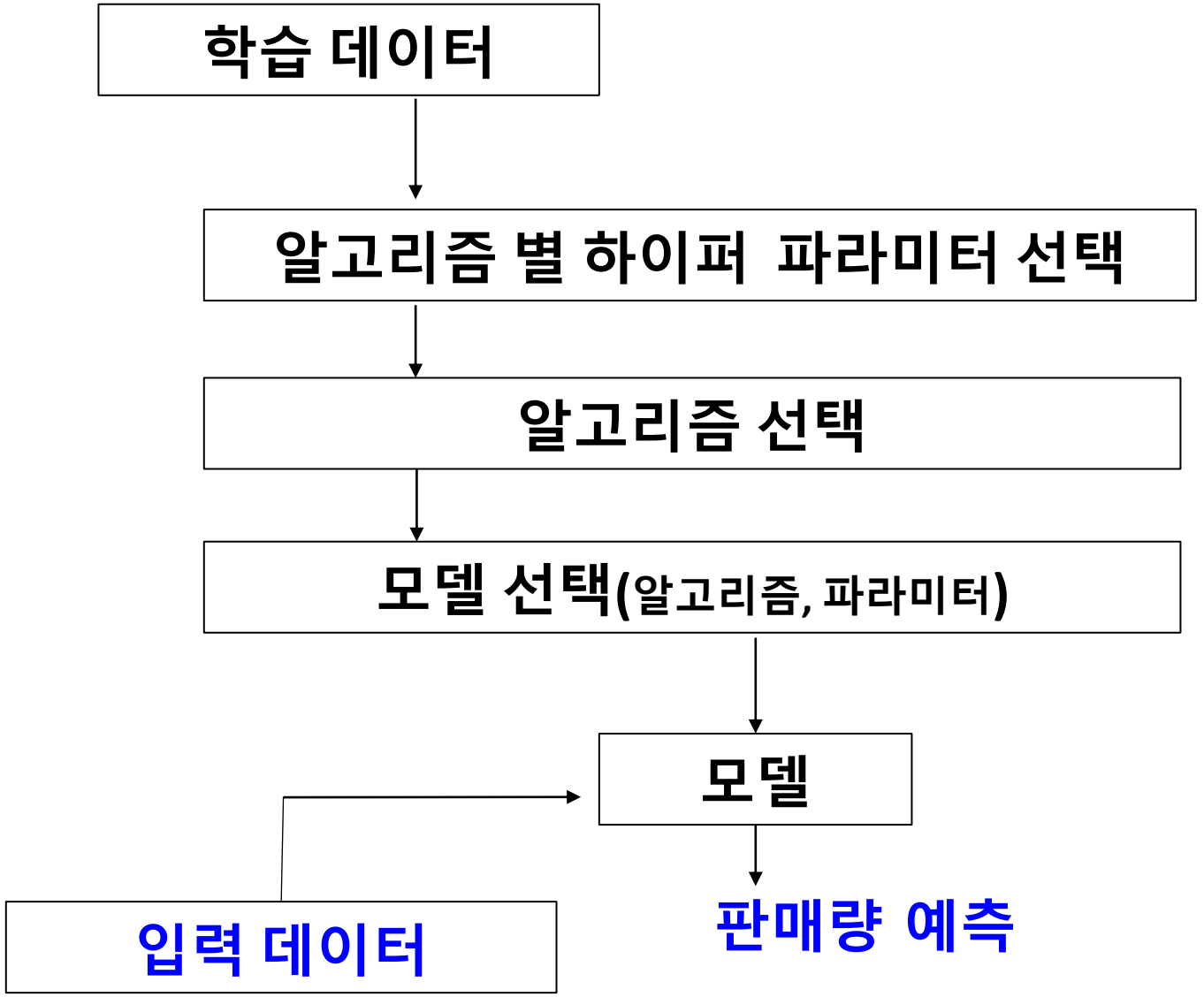
날짜	요일	온도	판매량

입력
독립변수
특성
속성
Feature

출력 (target)
종속 변수

입력 데이터

날짜	요일	온도



예측 (prediction)


```
from sklearn.ensemble import AdaBoostClassifier

classifier = AdaBoostClassifier()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

머신러닝 알고리즘 리스트

인하공전 컴퓨터 정보 과

algorithm	library	sub-lib	회귀(regression)	분류(classifier)
Ridge	sklearn	linear_model	Ridge	RidgeClassifier
Lasso	sklearn	linear_model	Lasso	
Linear Regression	sklearn	linear model	LinearRegression	
Support vector	sklearn	svm	SVR	SVC
SGD	sklearn	linear_model	SGRRegressor	SGDClassifier
kneighbor	sklearn	neighbors	KNeighborsRegressor	KNeighborsClassifier
K means	sklearn	cluster		KMeans
NaiveBayes	sklearn	naïve_bayes		GaussianNB
DecisionTree	sklearn	tree	DecisionTreeRegressor	DecisionTreeClassifier
Bagging	sklearn	ensemble	BaggingRegressor	BaggingClassifier
RandomForst	sklearn	ensemble	RandomForestRegressor	RandomForestClassifier
Adaboost	sklearn	ensemble	AdaBoostRegressor	AdaBoostClassifier
GradientBoost	sklearn	ensemble	GradientBoostingRegressor	GradientBoostingClassifier
HistGradientBost	sklearn	ensemble	HistGradientBoostingRegressor	HistGradientBoostingClassifier
voting	sklearn	ensemble	VotingRegressor	VotingClassifier
stacking	sklearn	ensemble	StackingRegressor	StackingClassifier
xgboost	xgboost		XGBRegressor	XGBClassifier
lightgbm	lightgbm		LGBMRegressor	LGBMClassifier

과제 feature importance

인디안 당뇨병 예측

중요도 높은 feature	adaboost	XGboost	LGBM
1st			
2nd			

수고하셨습니다

jhmin@inhatc.ac.kr