KU x depa

MACHINE LEARNING

Training Course 2022

Ensemble Method

DR. SETHAVIDH GERTPHOL

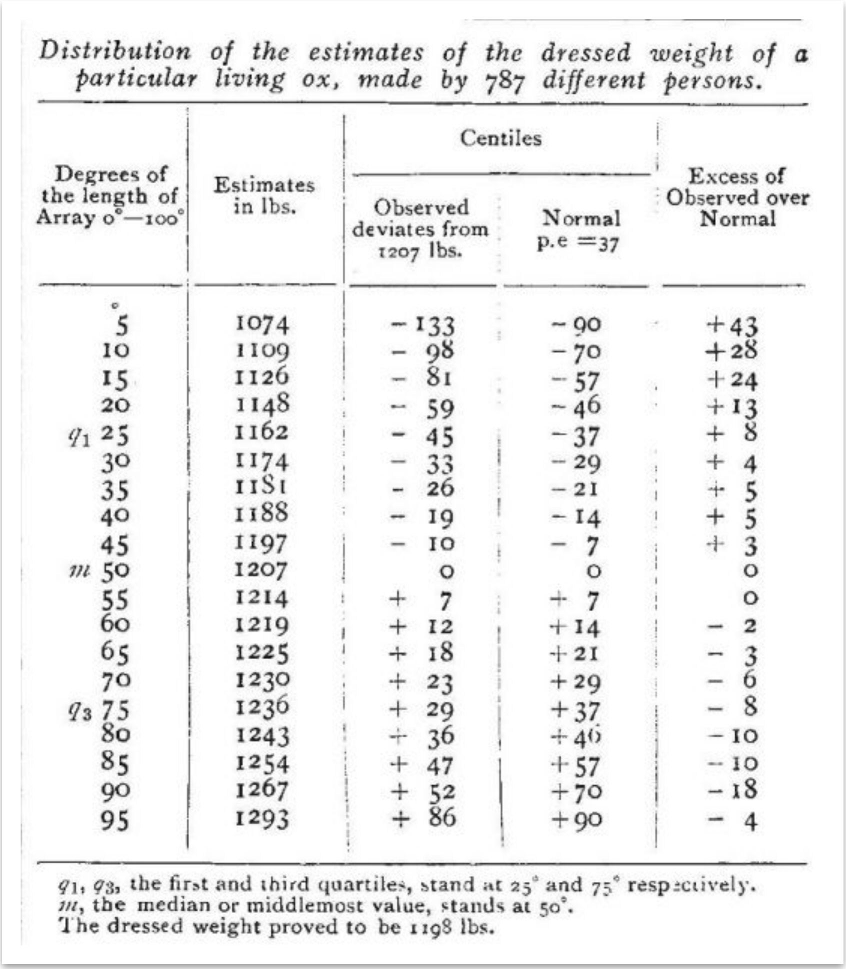
KU KASETSART UNIVERSITY depa

Today's Outline

- Ensemble Method
- Voting Classifier
- Bagging (Random Forest)
- Boosting
    - Adaboost
    - Gradient Boosting
- Stacking

Wisdom of the crowd

- Sir Francis Galton observed that an aggregated of estimates made by 787 persons about weight of an ox had only 1% error from the true wright.
- It may be better to aggregate predictions from several models instead of building a perfect model, even though some of them may be very bad.
- This is called Ensemble Method

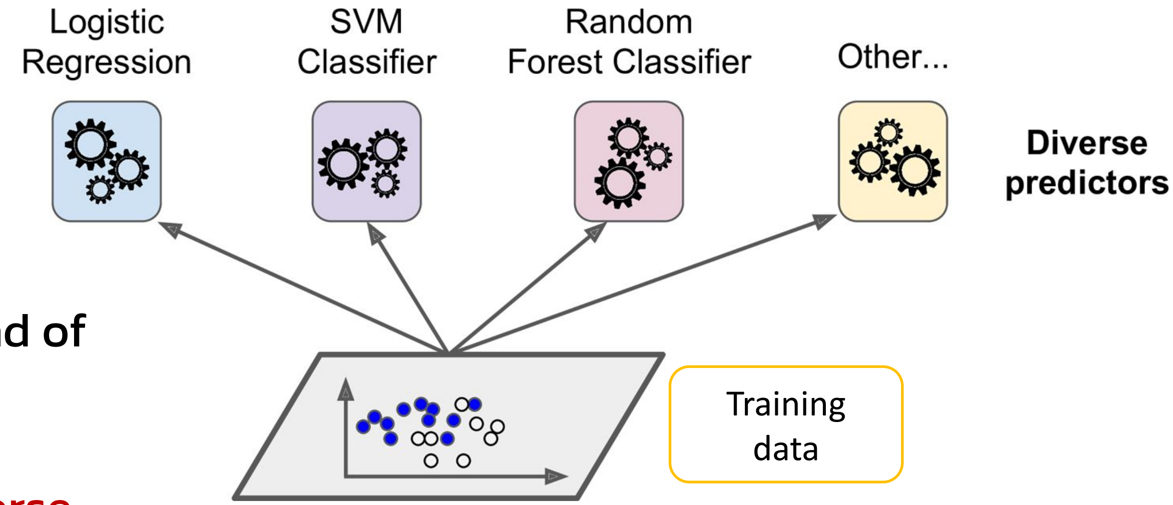


Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles		Excess of Observed over Normal
		Observed deviates from 1207 lbs.	Normal p.e =37	
5°	1074	−133	−90	+43
10	1109	−98	−70	+28
15	1126	−81	−57	+24
20	1148	−59	−46	+13
q_1 25	1162	−45	−37	+8
30	1174	−33	−29	+4
35	1181	−26	−21	+5
40	1188	−19	−14	+5
45	1197	−10	−7	+3
m 50	1207	0	0	0
55	1214	+7	+7	0
60	1219	+12	+14	−2
65	1225	+18	+21	−3
70	1230	+23	+29	−6
q_3 75	1236	+29	+37	−8
80	1243	+36	+46	−10
85	1254	+47	+57	−10
90	1267	+52	+70	−18
95	1293	+86	+90	−4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

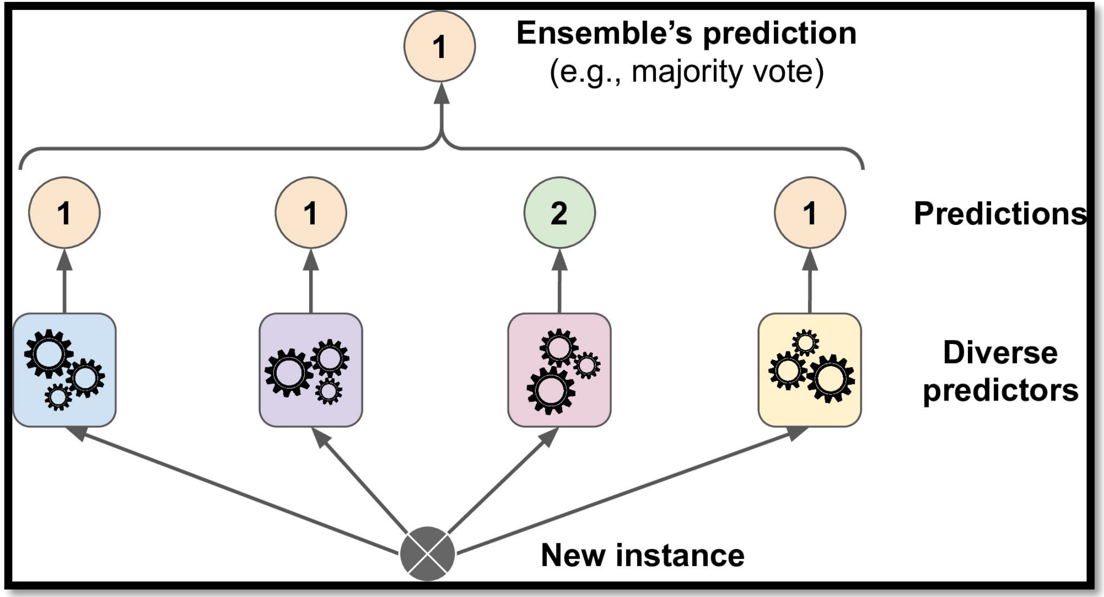
Ensemble Method



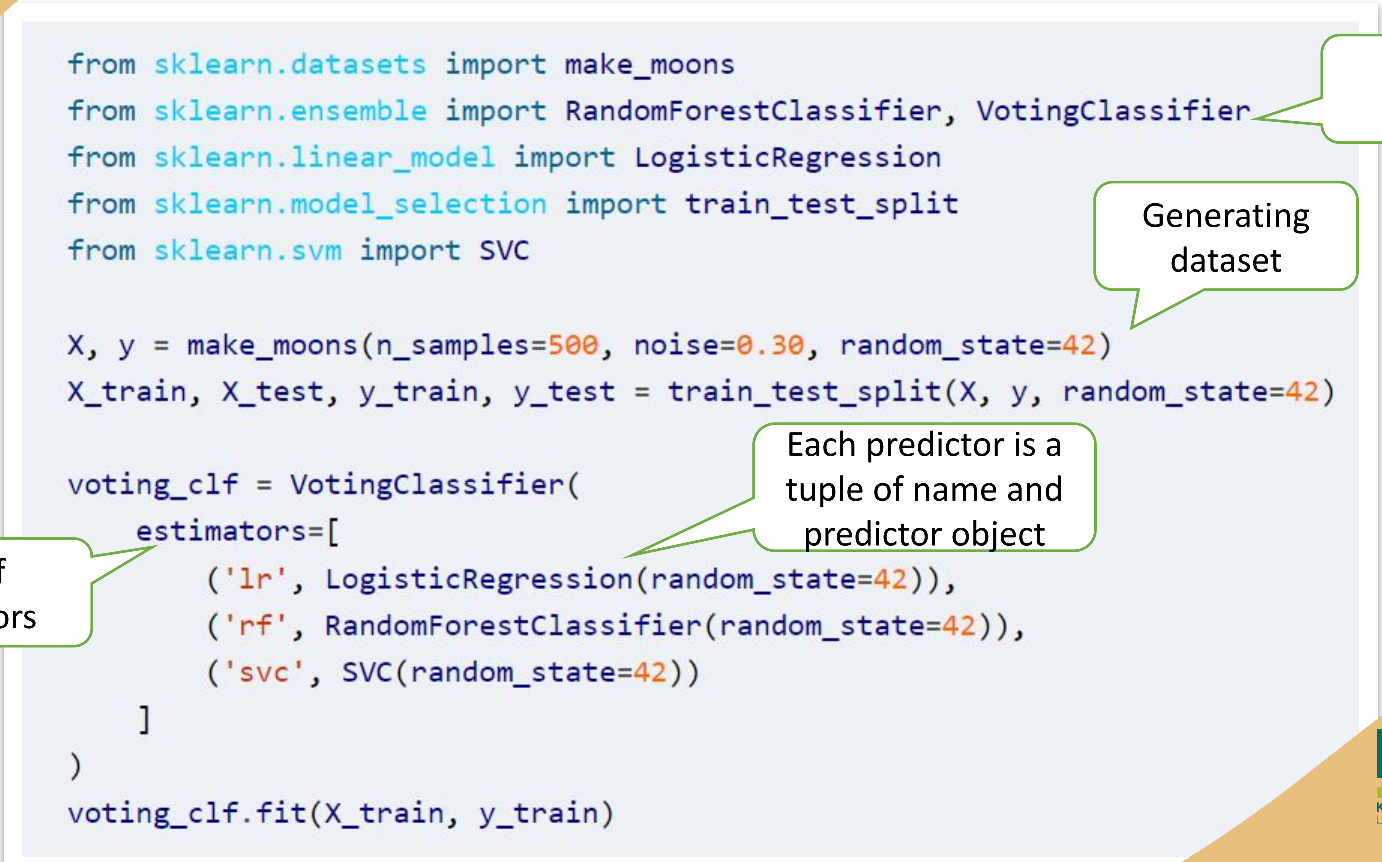
- Usually used at the end of project when we have several good models
- Models should be diverse
  - Using different algorithms or
  - Using different data

Voting Classifier

- A simple majority vote classifier
- Ensemble's prediction follows the majority vote of all classifiers



In Sklearn



```
from sklearn.datasets import make_moons
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

X, y = make_moons(n_samples=500, noise=0.30, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

voting_clf = VotingClassifier(
    estimators=[
        ('lr', LogisticRegression(random_state=42)),
        ('rf', RandomForestClassifier(random_state=42)),
        ('svc', SVC(random_state=42))
    ]
)
voting_clf.fit(X_train, y_train)
```

import

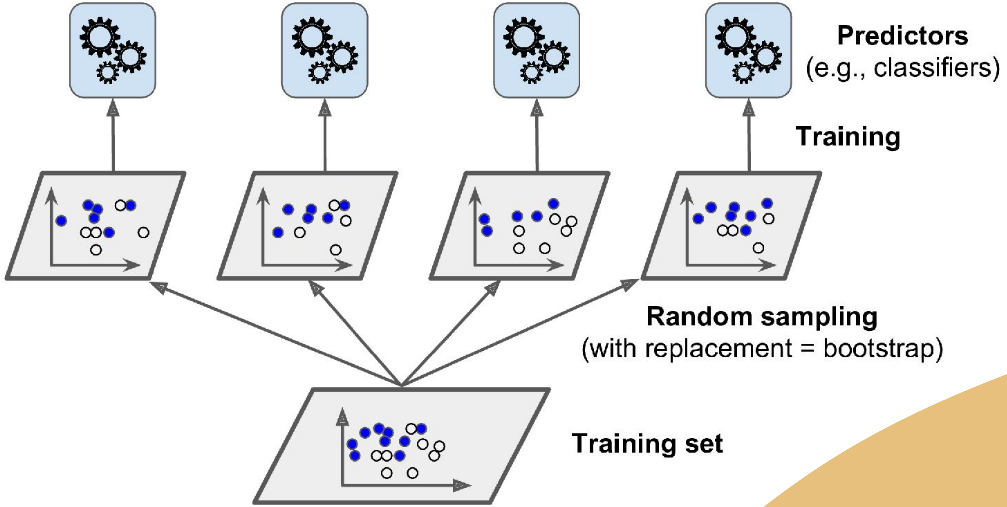
Generating dataset

Each predictor is a tuple of name and predictor object

List of predictors

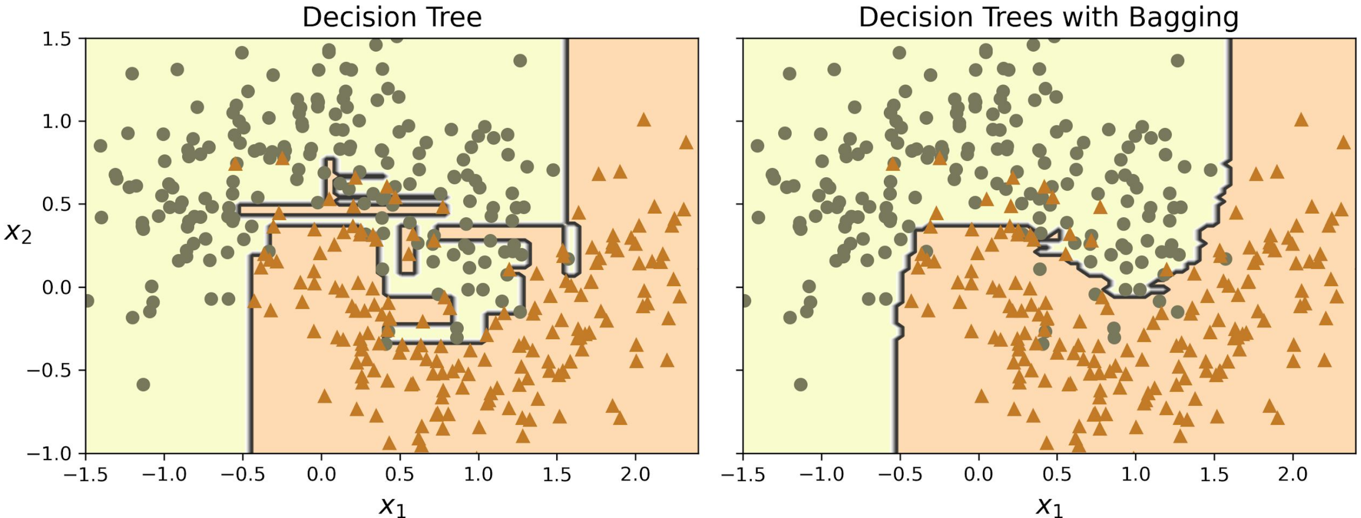
Bagging and Pasting

- One way to have diverse predictors is to use the same algorithm on different training data
- Use sampling to build different training set for each predictor
- Bagging (bootstrap aggregation) is sampling with replacement
- Pasting is sampling without replacement

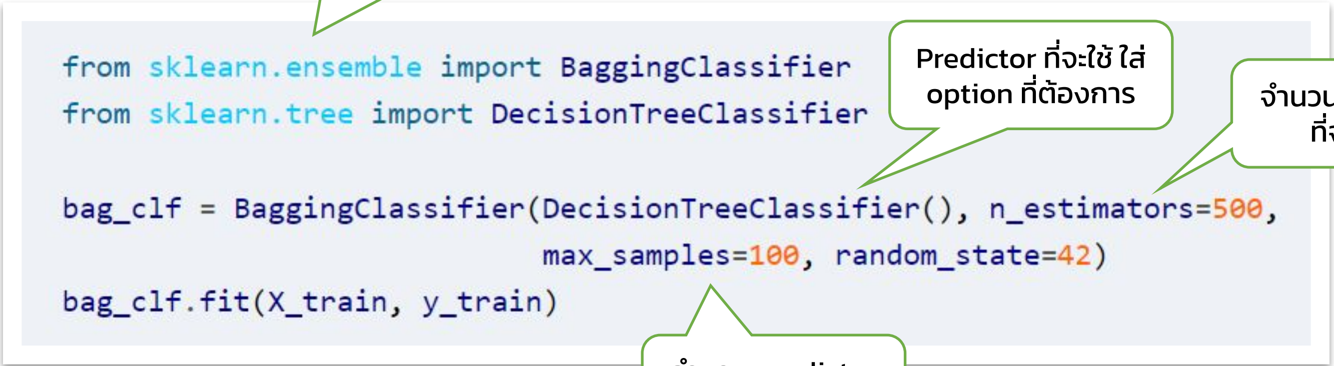


Bagging

- Generally better performance and more preferred than pasting
- If sampling size of a predictor is equal to training size, one predictors on average will see only 67% of the training data (33% are duplicates)
- Creates more bias on each predictor, but this bias make predictors more diverse
- Out-of-bag evaluation: use samples that are not included in a sample set as a test set of that predictor (no need for separate validation set)



In Sklearn



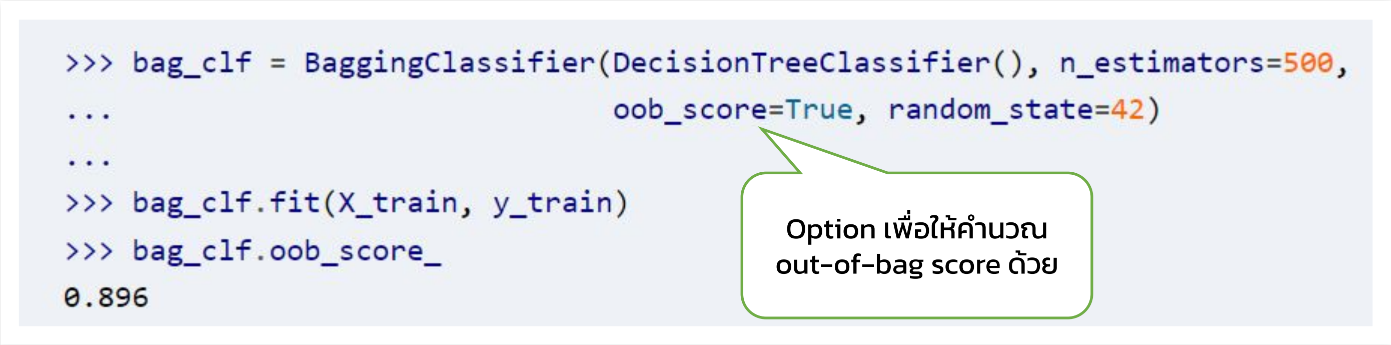
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=500,
                            max_samples=100, random_state=42)
bag_clf.fit(X_train, y_train)
```

import

Predictor ที่จะใช้ ใส่ option ที่ต้องการ

จำนวน predictor ที่จะสร้าง

จำนวน predictor ที่จะสร้าง



```
>>> bag_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=500,
...                             oob_score=True, random_state=42)
...
>>> bag_clf.fit(X_train, y_train)
>>> bag_clf.oob_score_
0.896
```

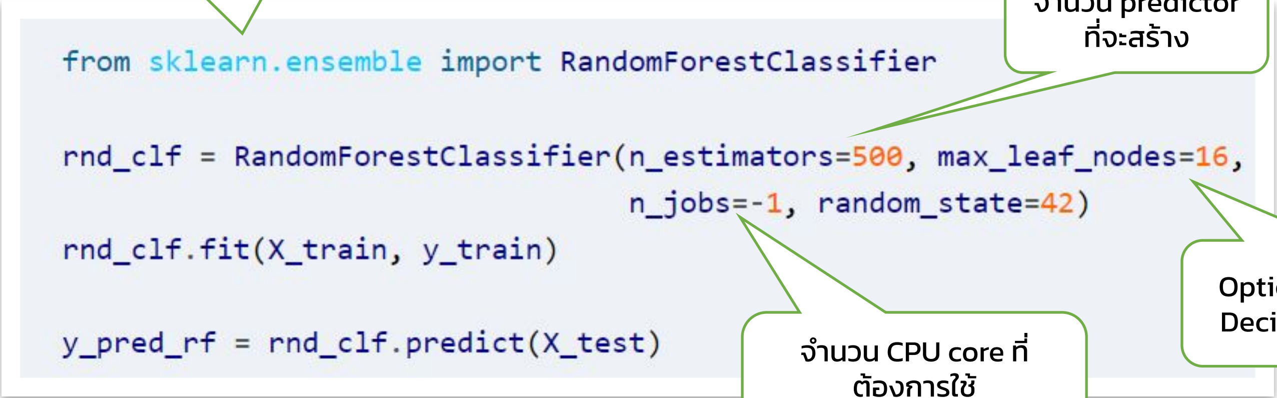
Option เพื่อให้คำนวณ out-of-bag score ด้วย

Random Forest

- An ensemble of Decision Tree built using bagging or pasting, usually with sample size equal to training set size
- Introduce additional randomness by selecting the best feature among a random subset of features to split each node
- Feature importance is the weight average of reduction in impurity of a node when using that feature to split it
    - Average across all trees
    - Weighted by the number of samples associated with that node

In Sklearn

import



```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16,
                                 n_jobs=-1, random_state=42)
rnd_clf.fit(X_train, y_train)

y_pred_rf = rnd_clf.predict(X_test)
```

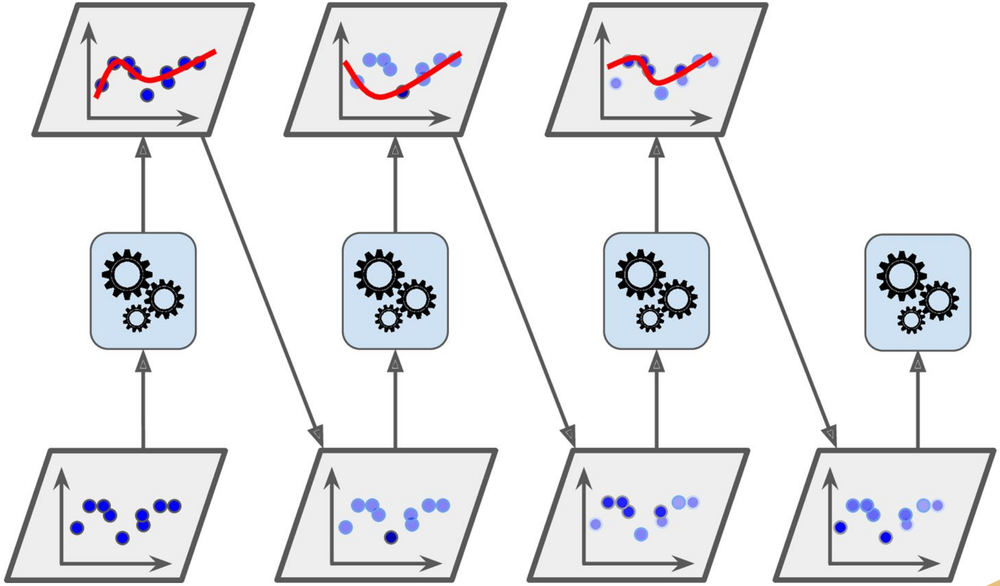
จำนวน predictor ที่จะสร้าง

Option อื่นใช้เหมือนของ DecisionTreeClassifier

จำนวน CPU core ที่ต้องการใช้
-1 คือใช้ทุก core

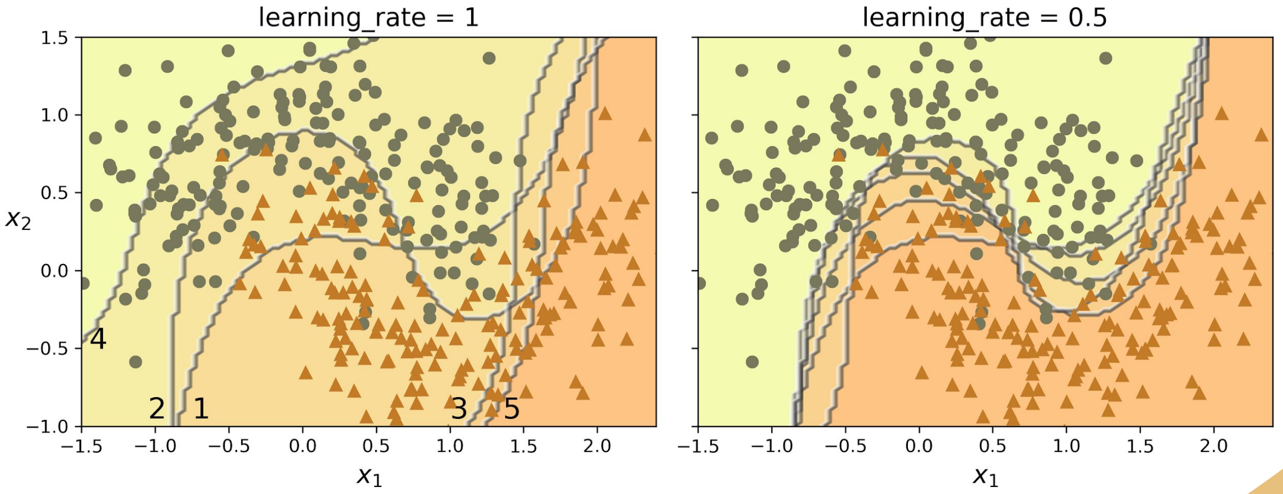
Boosting

- An ensemble method that trains predictors sequentially, with later ones trying to improve on the previous ones
- AdaBoost
  - Train 1 predictor, see which training instances are predicted wrong
  - Increase weight of those instances, train another predictors
  - Keep doing until reaching the number of predictors required



AdaBoost

- Learning rate: affect how much weight of wrong instances got boosted
- Predictor's weight: depends on how many instances it got right
- Final prediction is the weighted average of all predictors' prediction



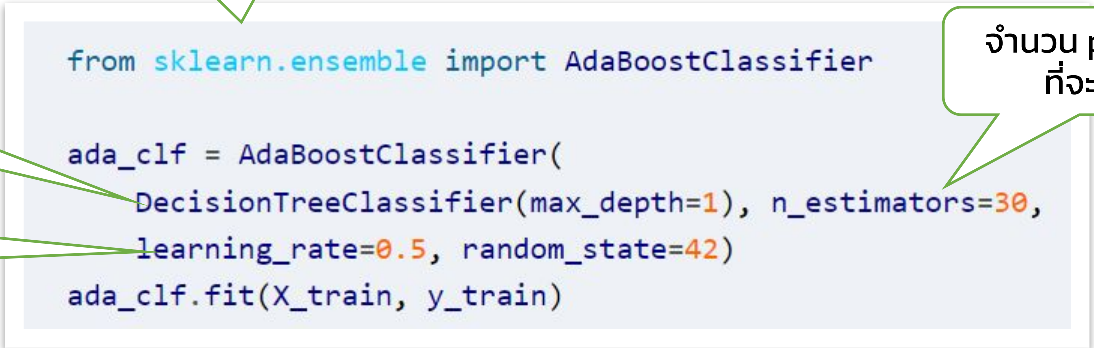
In Sklearn

import

Predictor ที่จะใช้

Learning rate

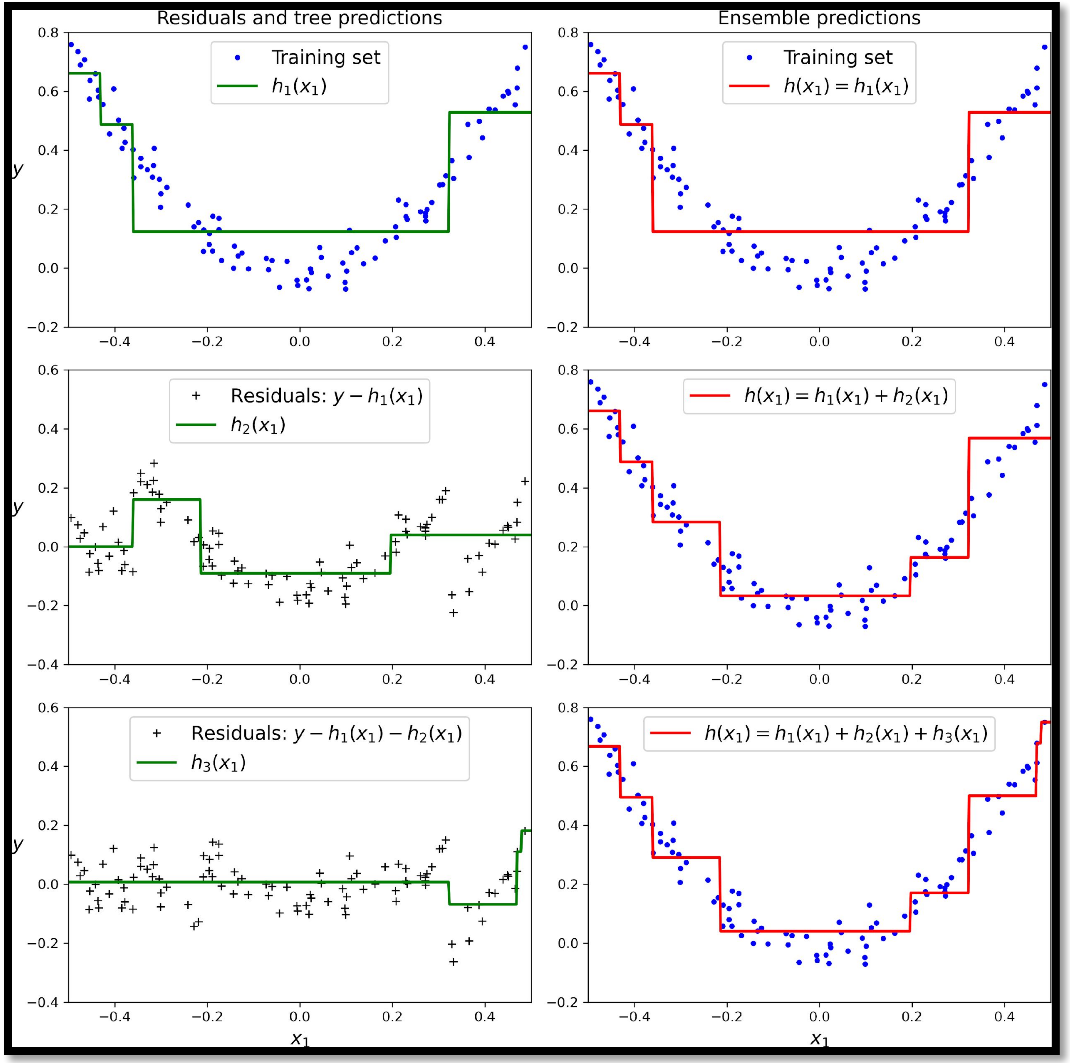
จำนวน predictor
ที่จะสร้าง



```
from sklearn.ensemble import AdaBoostClassifier

ada_clf = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1), n_estimators=30,
    learning_rate=0.5, random_state=42)
ada_clf.fit(X_train, y_train)
```

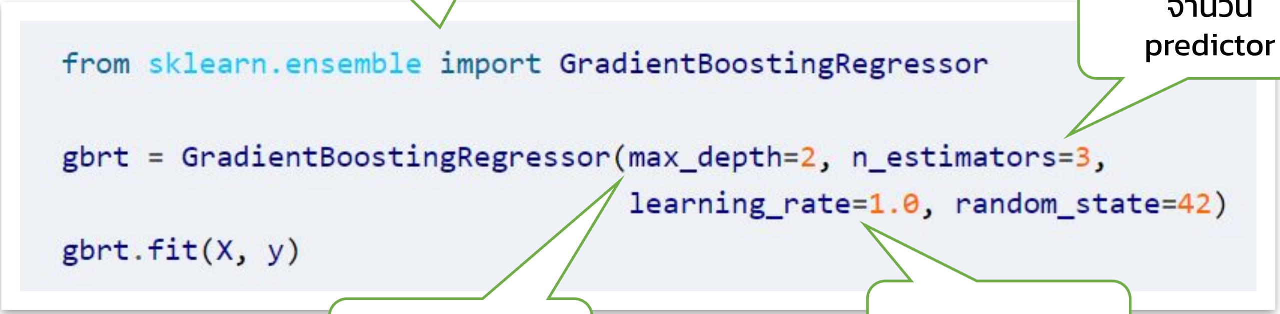
Gradient Boosting

- Train predictors sequentially, but later predictors try to predict residual errors from previous predictors
- Learning rate: how much contribution each predictor gives
  - If low, needs more predictors



In Sklearn

import

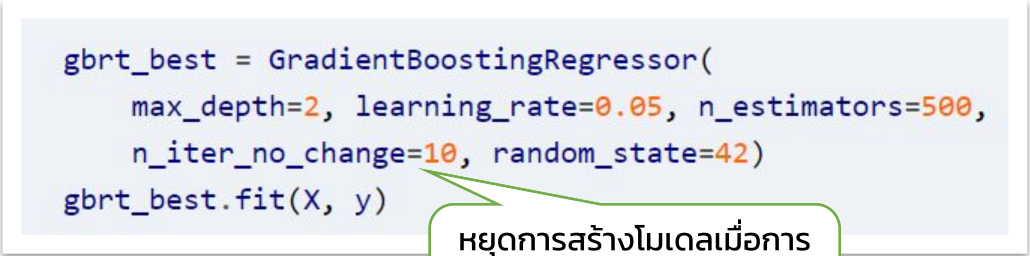


```
from sklearn.ensemble import GradientBoostingRegressor

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3,
                                 learning_rate=1.0, random_state=42)
gbrt.fit(X, y)
```

จำนวน
predictor

Option ของ
decision tree

Learning rate

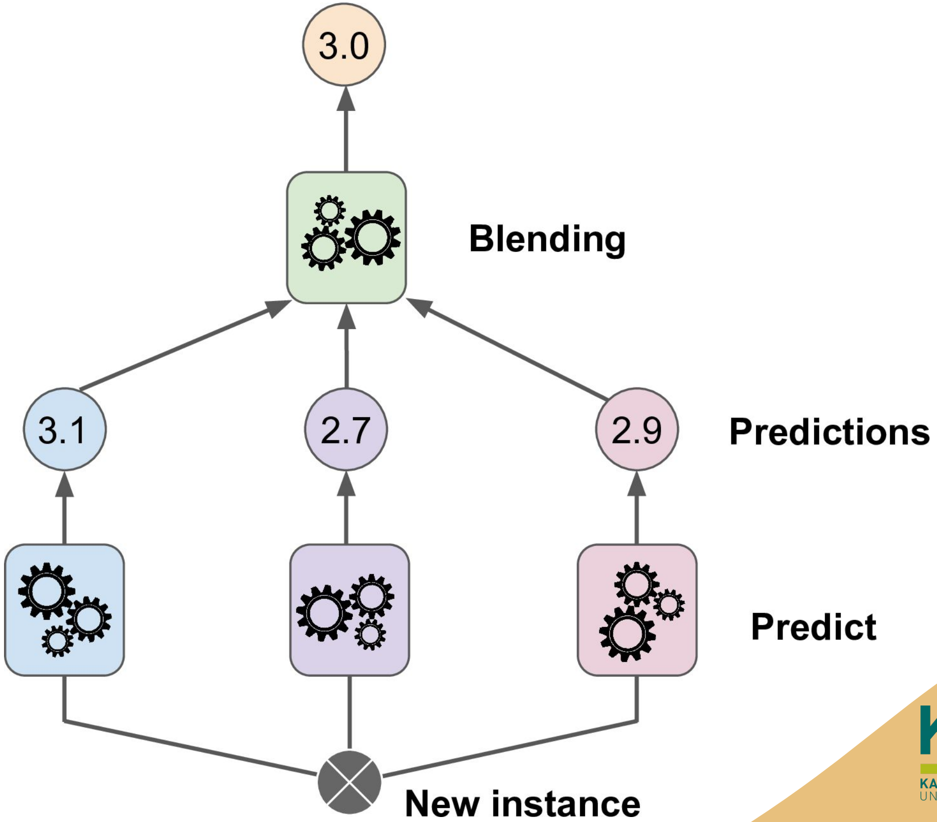


```
gbrt_best = GradientBoostingRegressor(
    max_depth=2, learning_rate=0.05, n_estimators=500,
    n_iter_no_change=10, random_state=42)
gbrt_best.fit(X, y)
```

หยุดการสร้างโมเดลเมื่อการ
ทำนายไม่เปลี่ยนแปลงเท่ากับ
จำนวนรอบที่ระบุ

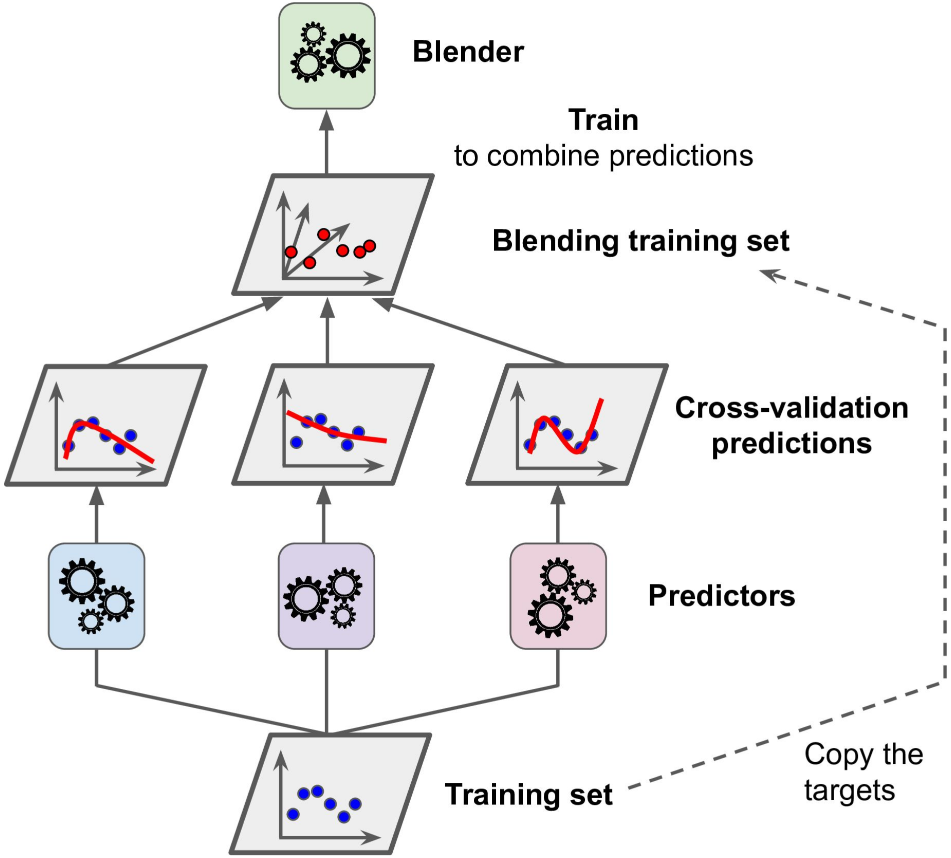
Stacking (stacked generalization)

- Similar to Voting, but train a blender or meta learner to aggregate predictions of other models

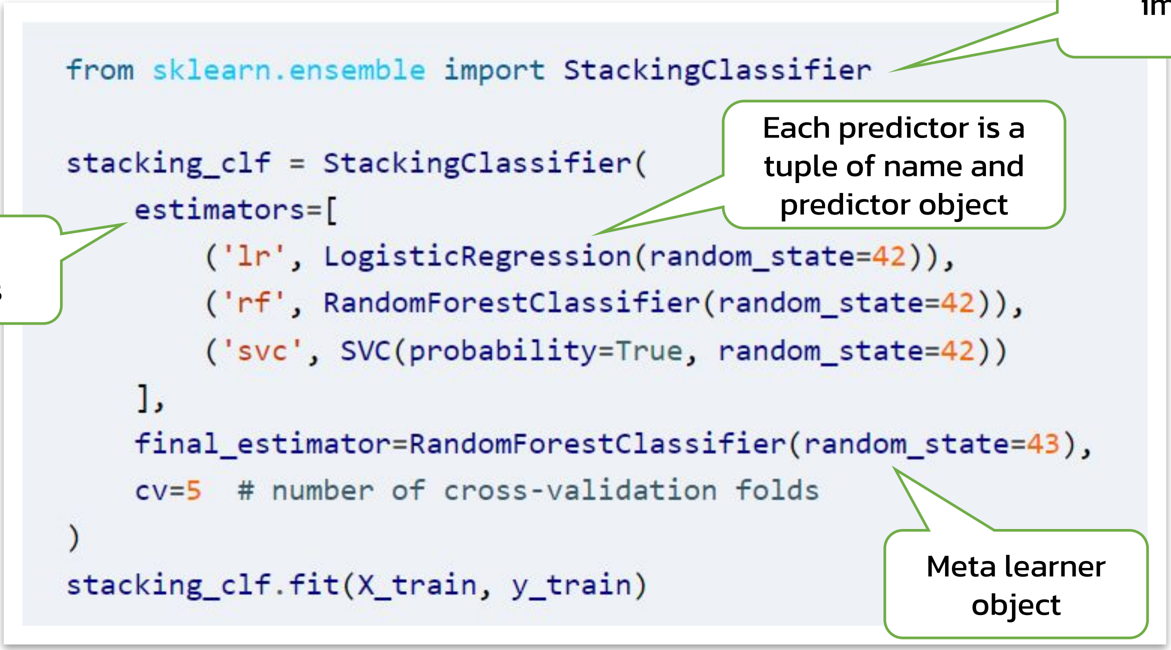


Training a meta learner

- Feature: prediction of each instance made by each predictor (can be obtained using cross-validation)
- Label: the truth value of each instance



In Sklearn



```
from sklearn.ensemble import StackingClassifier

stacking_clf = StackingClassifier(
    estimators=[
        ('lr', LogisticRegression(random_state=42)),
        ('rf', RandomForestClassifier(random_state=42)),
        ('svc', SVC(probability=True, random_state=42))
    ],
    final_estimator=RandomForestClassifier(random_state=43),
    cv=5  # number of cross-validation folds
)
stacking_clf.fit(X_train, y_train)
```

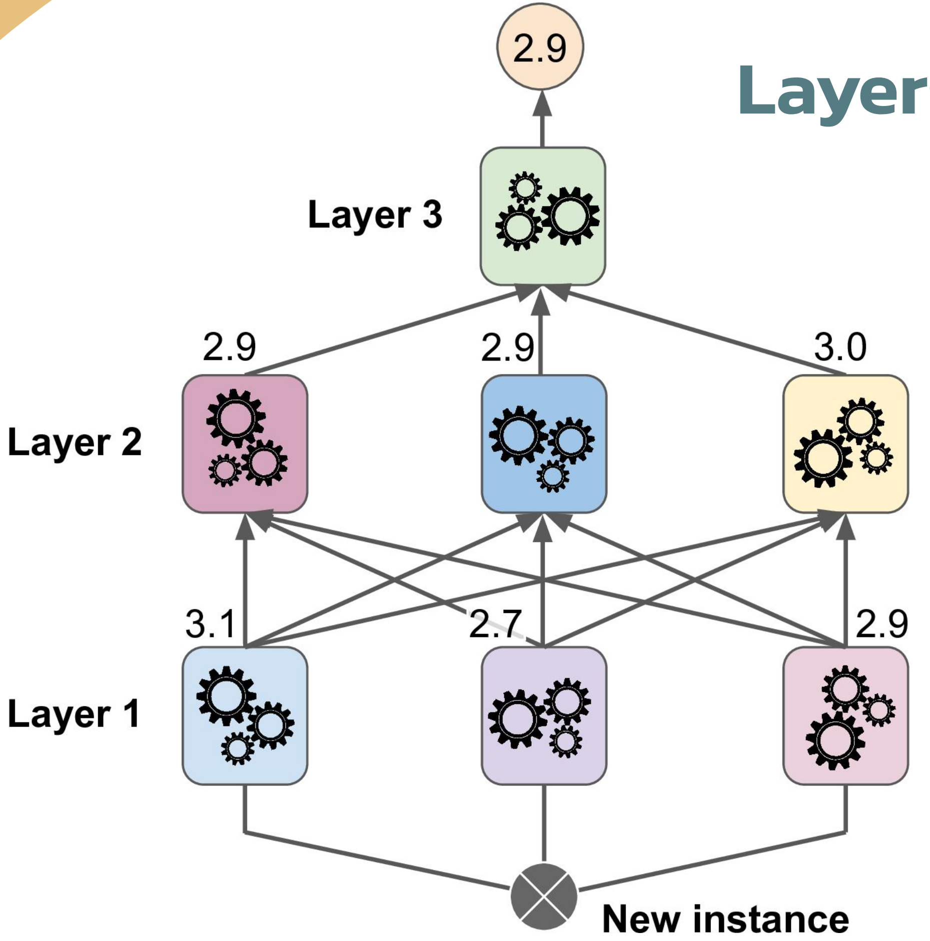
import

Each predictor is a tuple of name and predictor object

List of predictors

Meta learner object

Layers of Blenders



References

- GALTON, F. "Vox Populi" . Nature 75, 450–451 (1907). https://doi.org/10.1038/075450a0
- Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn and TensorFlow", O'Reilly Media, Inc., March 2017.