Bagging

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

Ensemble Methods

Voting Classifier

- same training set,
- $\neq$ algorithms.

다른 알고리즘을 사용하여 동일한 train set에 맞는 model의 앙상블
최종 예측이 다수결에 의해 얻어짐

Bagging

- one algorithm,
- $\neq$ subsets of the training set.

하나의 알고리즘
그러나 이러한 모델은 전체 train set에서 학습되지 않음
대신 각 모델은 data의 다른 하위 집합에 대해 학습

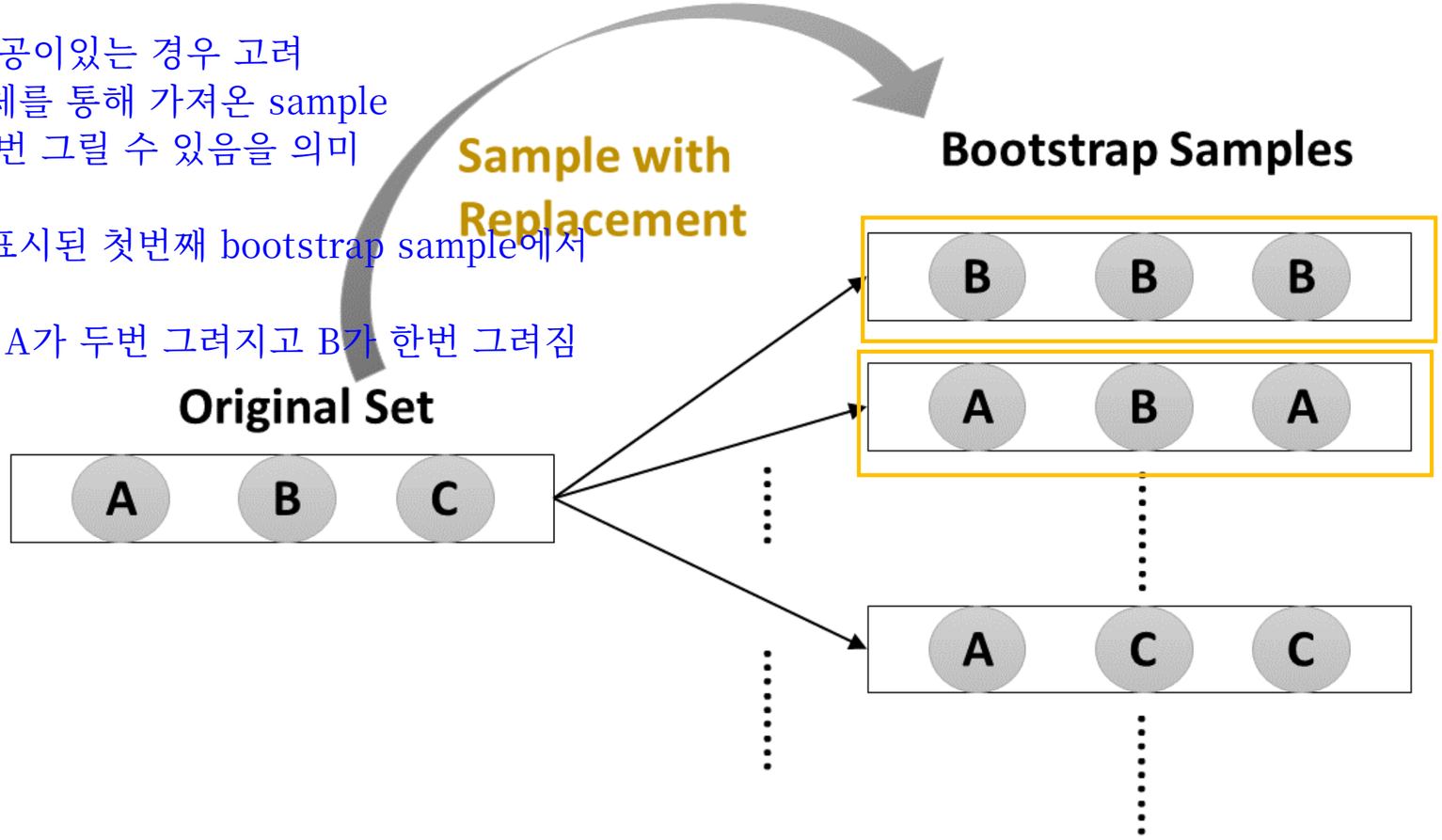
Bagging

- Bagging: Bootstrap Aggregation.
- Uses a technique known as the bootstrap.
- Reduces variance of individual models in the ensemble.

Bootstrap

A, B, C로 표시된 3개의 공이있는 경우 고려
bootstrap sample은 교체를 통해 가져온 sample
교체하면 모든 공을 여러번 그릴 수 있음을 의미

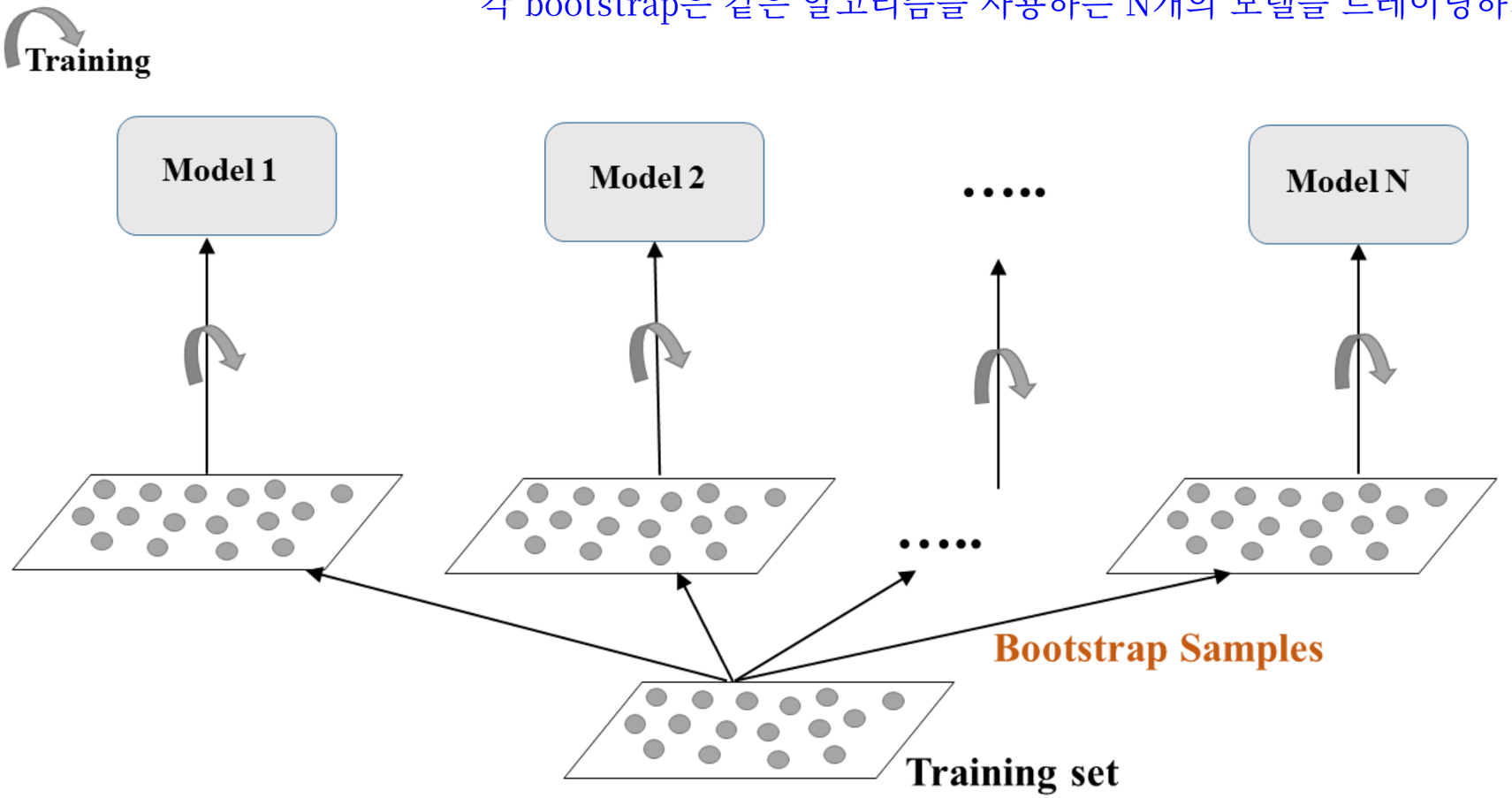
ex) 여기 다이어그램에 표시된 첫번째 bootstrap sample에서
B는 원래 3번 그려짐
두번째 bootstrap에서는 A가 두번 그려지고 B가 한번 그려짐



Bagging: Training

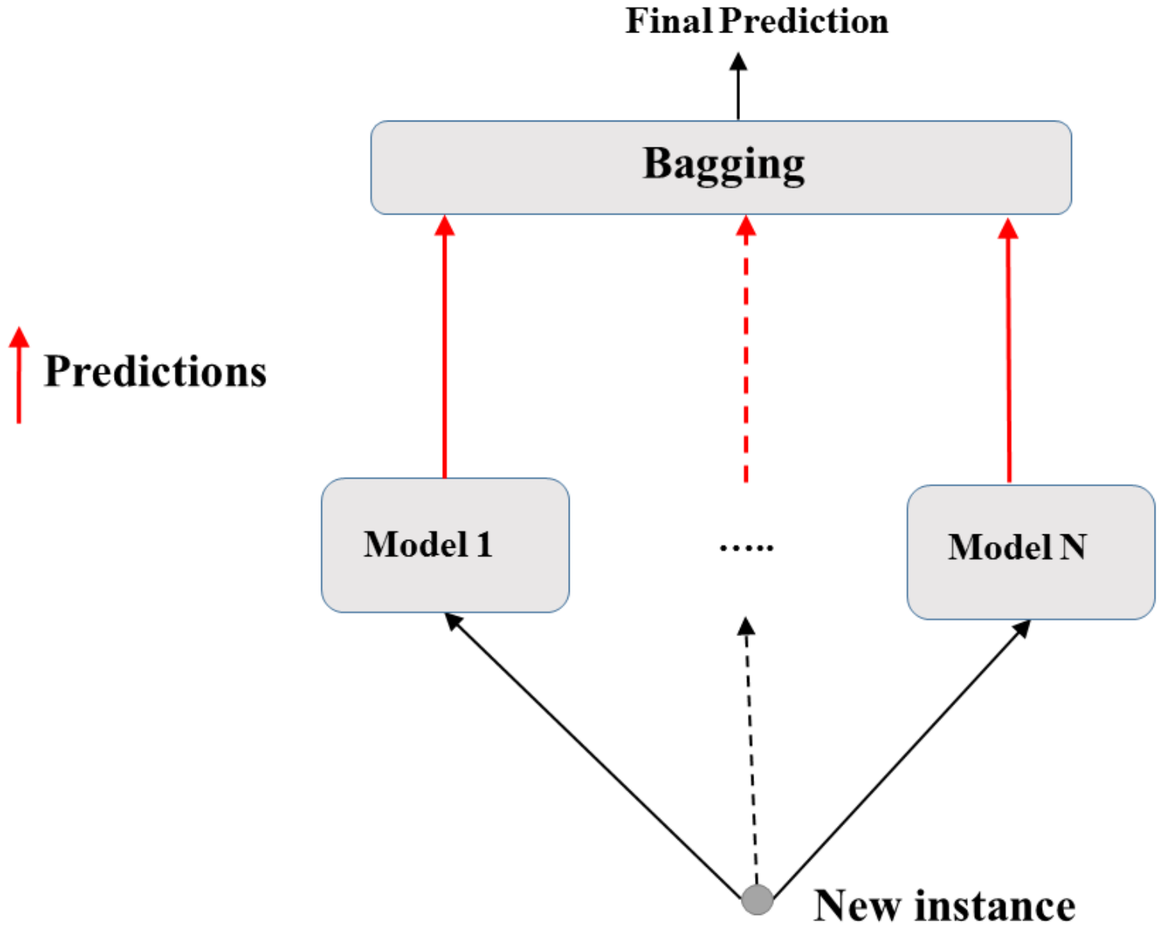
실제로 train 단계에서 bagging은
train set에서 N개의 다른 bootstrap sample을 그리는 것을 구성

각 bootstrap은 같은 알고리즘을 사용하는 N개의 모델을 트레이닝하기 위해 사용



Bagging: Prediction

새로운 instance가 bagging ensemble을 형성하는 다른 model에 공급되면
각 model은 예측을 출력
meta - model은 이러한 예측을 수집하고 문제의 성격에 따라 최종 예측을 출력



Bagging: Classification & Regression

Classification:

- Aggregates predictions by majority voting.
- `BaggingClassifier` in scikit-learn.

분류에서 최종 예측은 과반수 투표로 얻음

Regression:

- Aggregates predictions through averaging.
- `BaggingRegressor` in scikit-learn.

회귀 분석에서 최종 예측은 앙상블을 구성하는 개별 모형에 의해 이루어진 예측의 평균

Bagging Classifier in sklearn (Breast-Cancer dataset)

```
# Import models and utility functions
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Set seed for reproducibility
SEED = 1

# Split data into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    stratify=y,
                                                    random_state=SEED)
```

```
# Instantiate a classification-tree 'dt'
dt = DecisionTreeClassifier(max_depth=4, min_samples_leaf=0.16, random_state=SEED)
# Instantiate a BaggingClassifier 'bc'
bc = BaggingClassifier(base_estimator=dt, n_estimators=300, n_jobs=-1)
# Fit 'bc' to the training set
bc.fit(X_train, y_train)
# Predict test set labels
y_pred = bc.predict(X_test)

# Evaluate and print test-set accuracy
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy of Bagging Classifier: {:.3f}'.format(accuracy))
```

```
Accuracy of Bagging Classifier: 0.936
```

dt를 동일한 train set에 대해 훈련시키면 88.9%의 test set accuracy가 나오는데
이는 bagging이 dt보다 얼마나 우수한지 보여

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Out Of Bag Evaluation

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

Bagging

- some instances may be sampled several times for one model,
- other instances may not be sampled at all.

반면에 다른 인스턴스는 전혀 샘플링되지 않을 수 있음

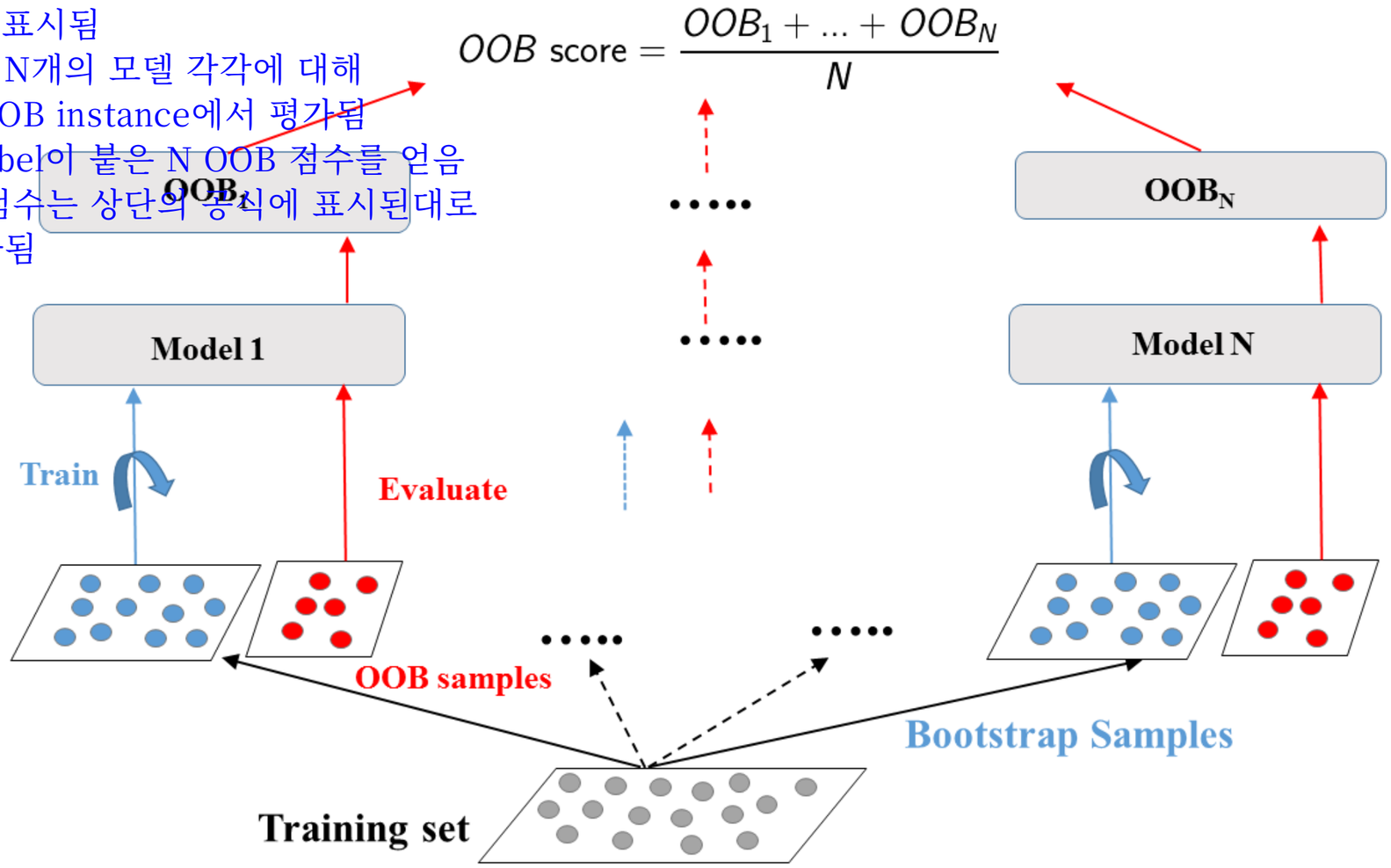
Out Of Bag (OOB) instances

- On average, for each model, 63% of the training instances are sampled.
- The remaining 37% constitute the OOB instances.

샘플링되지 않은 나머지 37%는 Out - of - bag 또는 OOB instance를 구성
OOB 인스턴스는 훈련 중에는 모델에 의해 표시되지 않기 때문에 교차 검증 없이 앙상블의 성능을 추정할 수 있음

OOB Evaluation

각 model에 대해 bootstrap instance는 파란색으로 표시되고
OOB instance는 빨간색으로 표시됨
그런 다음 앙상블을 구성하는 N개의 모델 각각에 대해
해당 bootstrap sample 및 OOB instance에서 평가됨
이것은 OOB1~OOBN으로 label이 붙은 N OOB 점수를 얻음
bagging ensemble의 OOB 점수는 상단의 공식에 표시된대로
N OOB 점수의 평균으로 평가됨



OOB Evaluation in sklearn (Breast Cancer Dataset)

```
# Import models and split utility function
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Set seed for reproducibility
SEED = 1

# Split data into 70% train and 30% test
X_train, X_test, y_train, y_test =  train_test_split(X, y, test_size= 0.3,
                                                     stratify= y,
                                                     random_state=SEED)
```

```
# Instantiate a classification-tree 'dt'
dt = DecisionTreeClassifier(max_depth=4,
                            min_samples_leaf=0.16,
                            random_state=SEED)
# Instantiate a BaggingClassifier 'bc'; set oob_score = True
bc = BaggingClassifier(base_estimator=dt, n_estimators=300,
                       oob_score=True, n_jobs=-1)
# Fit 'bc' to the training set
bc.fit(X_train, y_train)

# Predict the test set labels
y_pred = bc.predict(X_test)
```

잎당 샘플의 최소 비율 = 16%

훈련 후 bc의 OOB 정확도를 평가하기 위해 매개 변수 oob_score를 True로 설정하는 것
skikit - learn에서는 OOB 점수는 분류자의 정확도와 regressor의 r - 제곱 점수에 대응

```
# Evaluate test set accuracy
test_accuracy = accuracy_score(y_test, y_pred)
# Extract the OOB accuracy from 'bc'
oob_accuracy = bc.oob_score_

# Print test set accuracy
print('Test set accuracy: {:.3f}'.format(test_accuracy))
```

```
Test set accuracy: 0.936
```

```
# Print OOB accuracy
print('OOB accuracy: {:.3f}'.format(oob_accuracy))
```

```
OOB accuracy: 0.925
```

test set accuracy와 OOB accuracy는 정확히 같지는 않지만 비슷함.
이러한 결과는 OOB 평가가 교차 검증을 수행하지 않고 보이지 않는 데이터에 대한 bagged - ensemble의 성능 추정치를 얻기 위한 효율적인 기술이 될 수 있음을 보여줌

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Random Forests

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

Bagging

- Base estimator: Decision Tree, Logistic Regression, Neural Net, ... 어떤 모델이 될 수있음
- Each estimator is trained on a distinct bootstrap sample of the training set
- Estimators use all features for training and prediction

각 추정기는 사용 가능한모든 feature를 사용하여
train_set에서 추출한 별개의 bootstrap sample에서 훈려됨

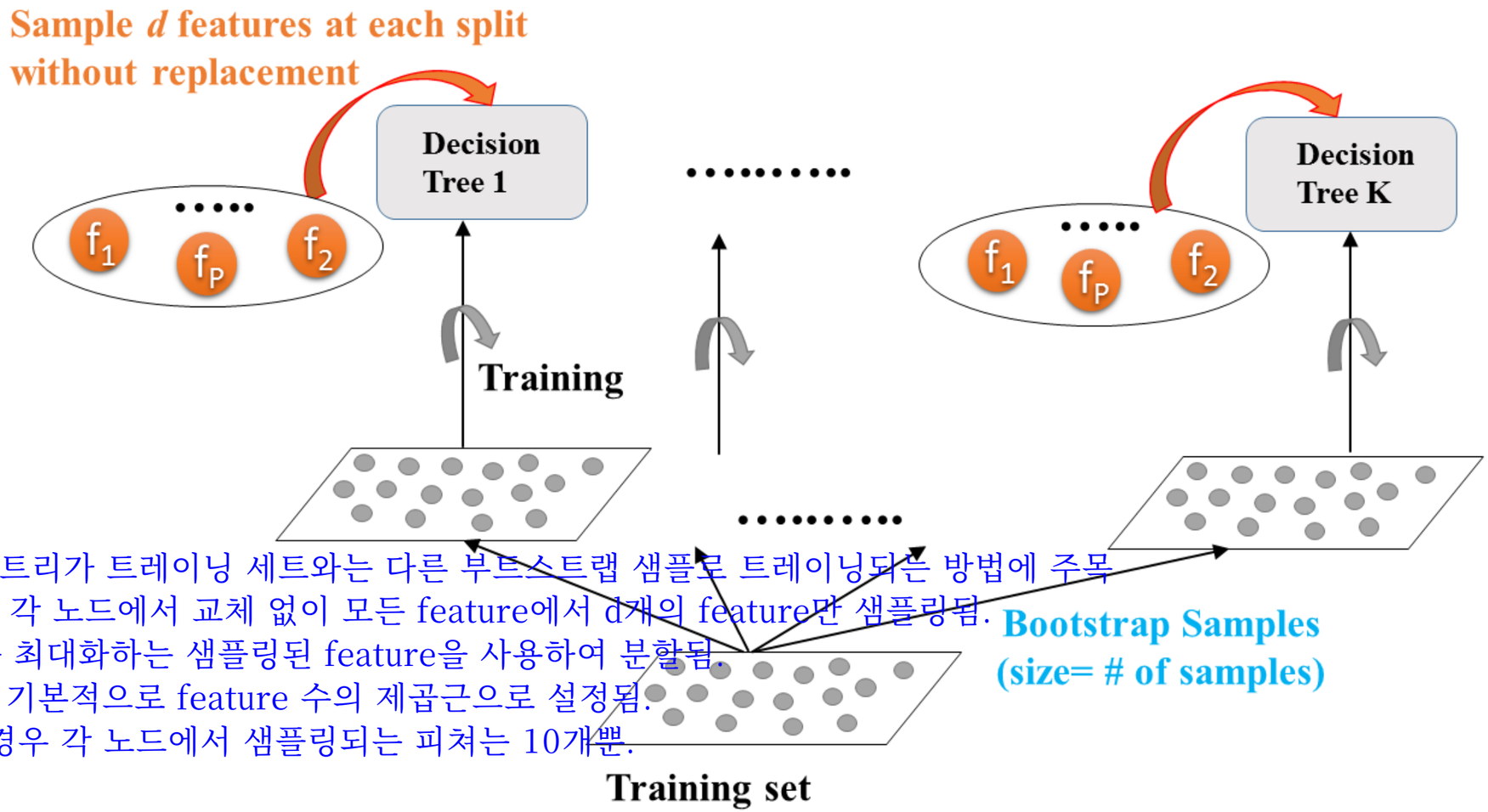
Further Diversity with Random Forests

- Base estimator: Decision Tree
- Each estimator is trained on a different bootstrap sample having the same size as the training set
- RF introduces further randomization in the training of individual trees
- d features are sampled at each node without replacement ($d < \text{total number of features}$)

RF는 각 기본 추정량을 훈련할 때 bagging보다 더 무작위 화를 도입함

각 tree가 학습되면 각 node에서 d개의 feature만 샘플링 할 수 있음
(여기서 d<total number of features)

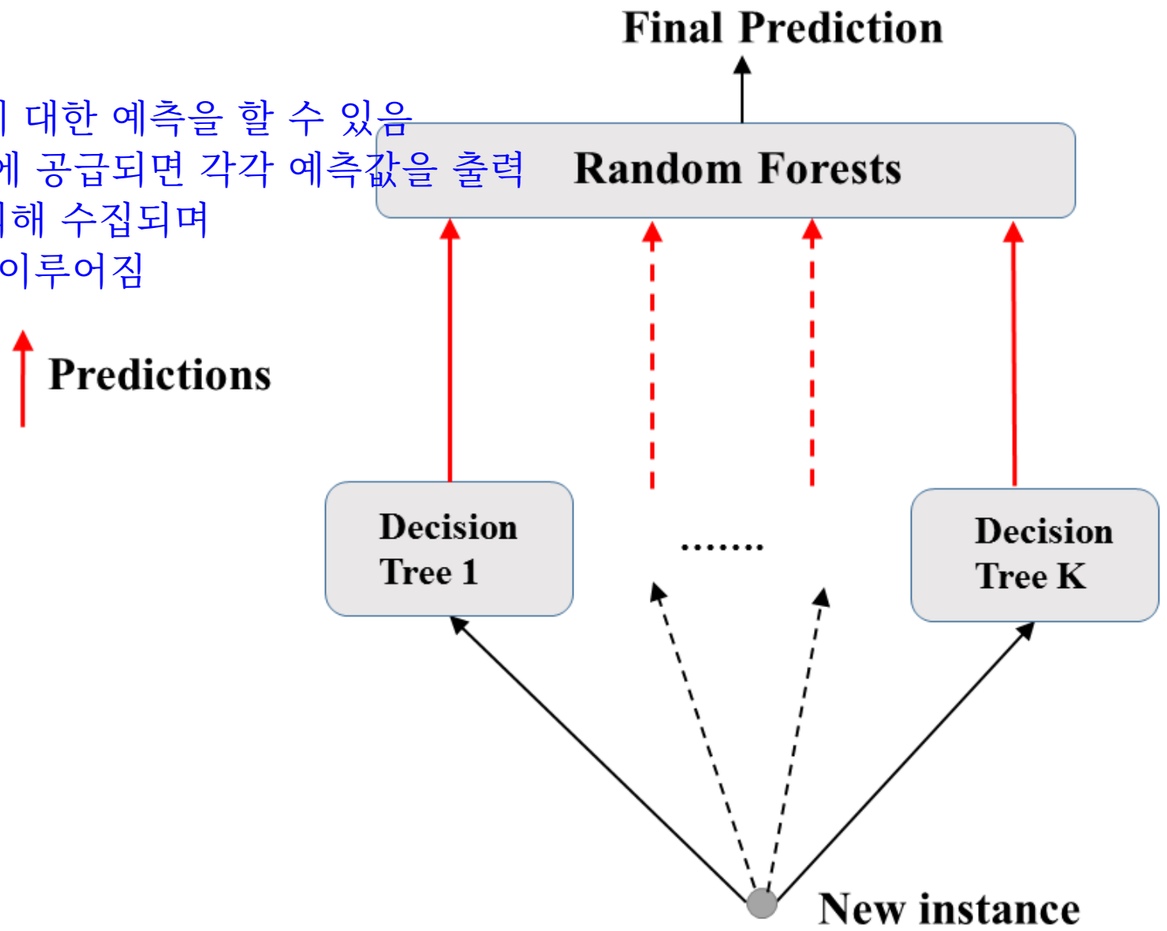
Random Forests: Training



앙상블을 형성하는 각 트리가 트레이닝 세트와는 다른 부트스트랩 샘플로 트레이닝되는 방법에 주목
또한 트리가 훈련될 때 각 노드에서 교체 없이 모든 feature에서 d개의 feature만 샘플링됨.
그런 다음 노드는 IG를 최대화하는 샘플링된 feature을 사용하여 분할됨.
skikit - learn d에서는 기본적으로 feature 수의 제곱근으로 설정됨.
ex) 피쳐가 100개일 경우 각 노드에서 샘플링되는 피쳐는 10개뿐.

Random Forests: Prediction

훈련을 받으면 새로운 instance에 대한 예측을 할 수 있음
새 instance가 다른 기본 추정기에 공급되면 각각 예측값을 출력
예측은 RF meta - classifier에 의해 수집되며
문제의 성격에 따라 최종 예측이 이루어짐



Random Forests: Classification & Regression

Classification:

- Aggregates predictions by majority voting (다수결)
- `RandomForestClassifier` in scikit-learn

Regression:

- Aggregates predictions through averaging
- `RandomForestRegressor` in scikit-learn

일반적으로 RF는 개별 tree보다 variance가 적음

Random Forests Regressor in sklearn (auto dataset)

```
# Basic imports
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
# Set seed for reproducibility
SEED = 1

# Split dataset into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=SEED)
```

```
# Instantiate a random forests regressor 'rf' 400 estimators
rf = RandomForestRegressor(n_estimators=400,
                           min_samples_leaf=0.12,
                           random_state=SEED)
# Fit 'rf' to the training set
rf.fit(X_train, y_train)
# Predict the test set labels 'y_pred'
y_pred = rf.predict(X_test)
```

```
# Evaluate the test set RMSE
rmse_test = MSE(y_test, y_pred)**(1/2)

# Print the test set RMSE
print('Test set RMSE of rf: {:.2f}'.format(rmse_test))
```

```
Test set RMSE of rf: 3.98
```

RF는 3.98의 test set RMS로 이 error는 단일 회귀 tree(4.43)에 의해 달성한 것보다 작음

Feature Importance

Tree-based methods: enable measuring the importance of each feature in prediction.

In `sklearn` :

- how much the tree nodes use a particular feature (weighted average) to reduce impurity
- accessed using the attribute `feature_importance_`

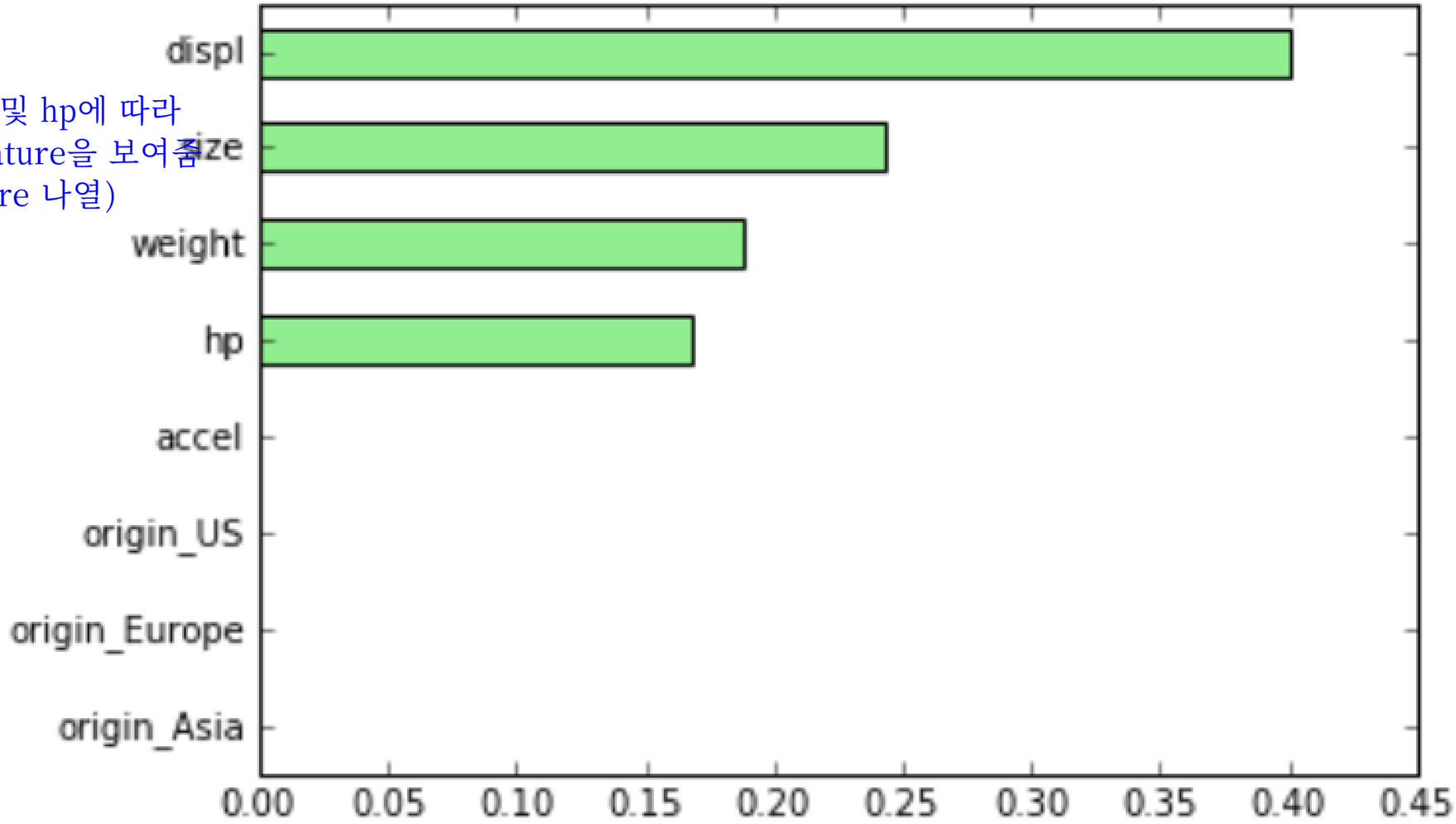
  - skikit - learn에서는 트리 노드가 불순물을 줄이기 위해 특정 기능을 얼마나 사용하는지를 측정하여 feature importance를 평가
  - feature의 중요도는 훈련 및 예측에서 해당 기능의 가중치를 나타내는 퍼센티지로 표시

Feature Importance in sklearn

```
import pandas as pd
import matplotlib.pyplot as plt

# Create a pd.Series of features importances
importances_rf = pd.Series(rf.feature_importances_, index = X.columns)

# Sort importances_rf
sorted_importances_rf = importances_rf.sort_values()

# Make a horizontal bar plot 수평 막대 그래프
sorted_importances_rf.plot(kind='barh', color='lightgreen'); plt.show()
```

Feature Importance in sklearn

결과
rf, displ, 크기, 무게 및 hp에 따라
가장 예측 가능한 feature을 보여줌
(중요한순대로 feature 나열)



Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON