Generalization Error

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

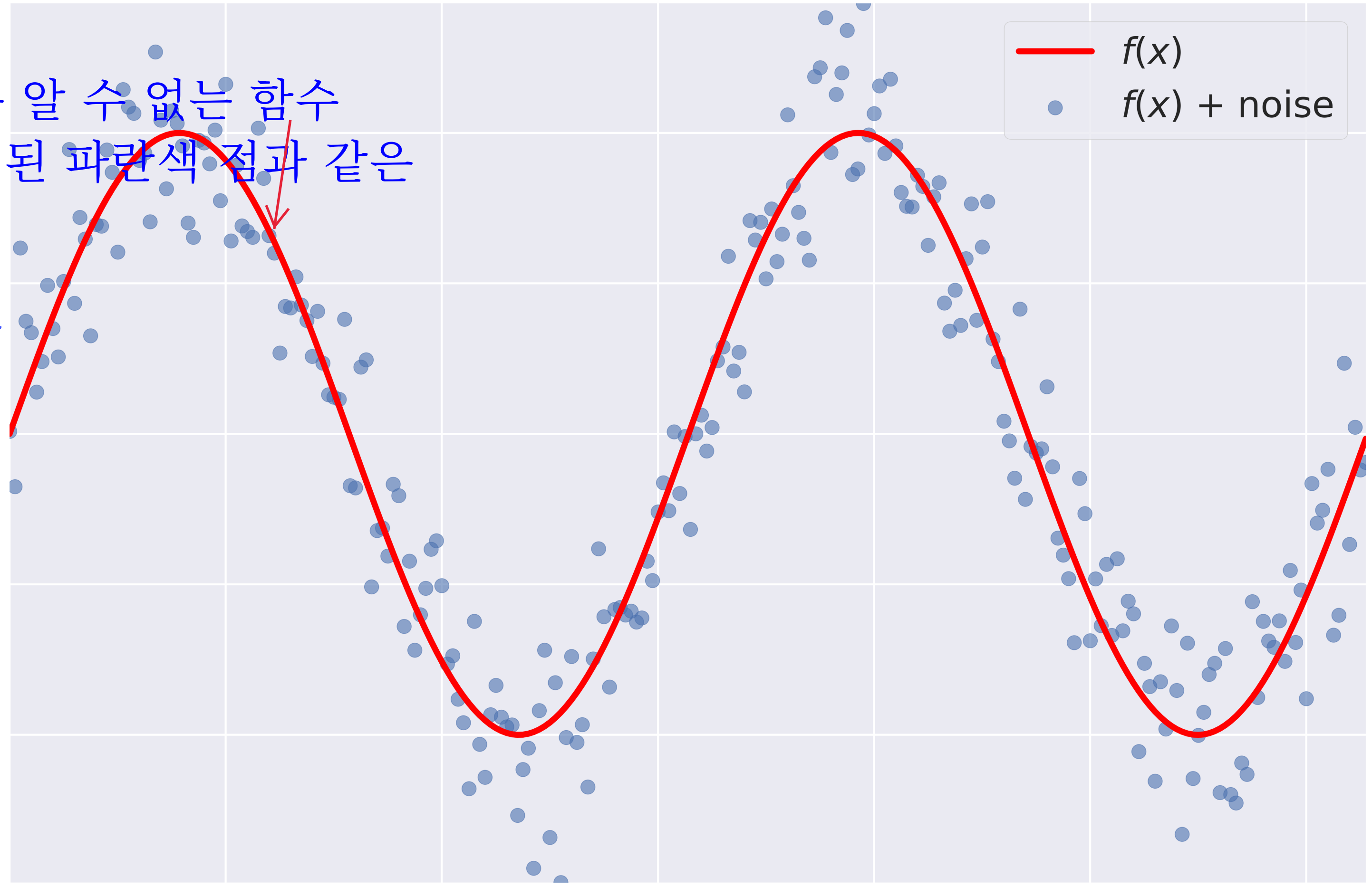
Elie Kawerk
Data Scientist

Supervised Learning - Under the Hood

- Supervised Learning: $y = f(x)$, f is unknown.

지도 학습에서는 feature와 label 사이에 매핑 f가 있다고 가정
=> y = f(x)로 표현할 수 있음

여기 빨간색으로 표시된 f는 결정하려는 알 수 없는 함수
실제 데이터 생성에는 항상 여기에 표시된 파란색 점과 같은
임의성 또는 noise가 수반됨



Goals of Supervised Learning

- Find a model $\hat{f}$ that best approximates f: $\hat{f} \approx f$
- $\hat{f}$ can be Logistic Regression, Decision Tree, Neural Network ...
- Discard noise as much as possible.
- **End goal:** $\hat{f}$ should achieve a low predictive error on unseen datasets.

목표는 f에 가장 가까운 모델 f_hat을 찾는 것임
f_hat을 훈련 할 때 가능한 많은 noise가 제거되도록해야함
결국 f_hat은 보이지 않은 데이터 세트에 대한 낮은 예측 오류를 달성해야함

Difficulties in Approximating f

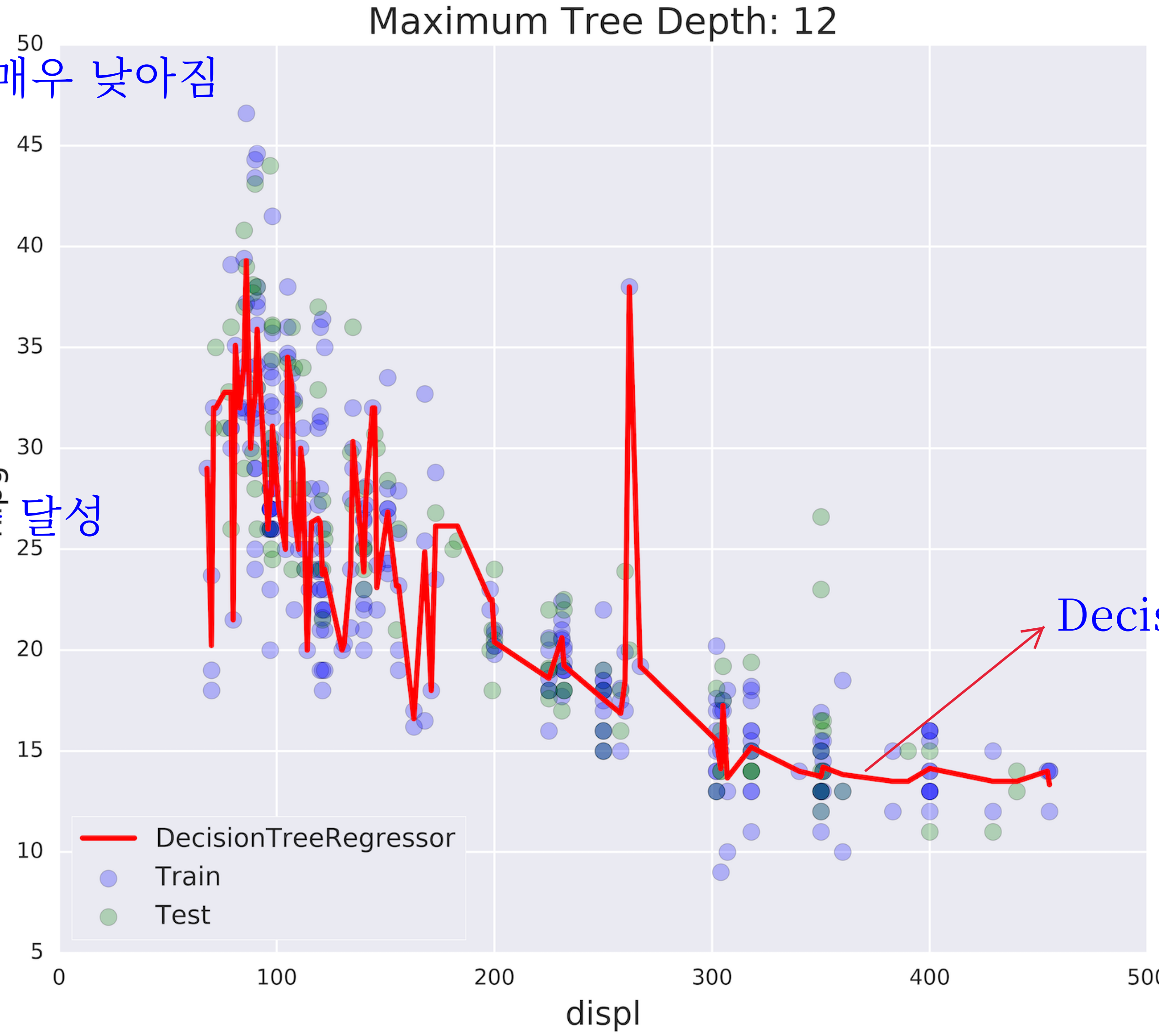
- **Overfitting:** $\hat{f}(x)$ fits the training set noise.
- **Underfitting:** $\hat{f}$ is not flexible enough to approximate f.

f를 근사화 할 때 두 가지 어려움이 발생
- overfitting : f_hat이 train_set의 noise에 맞을 때
- underfitting : f_hat이 f를 근사 할만큼 유연하지 않은 경우

Overfitting

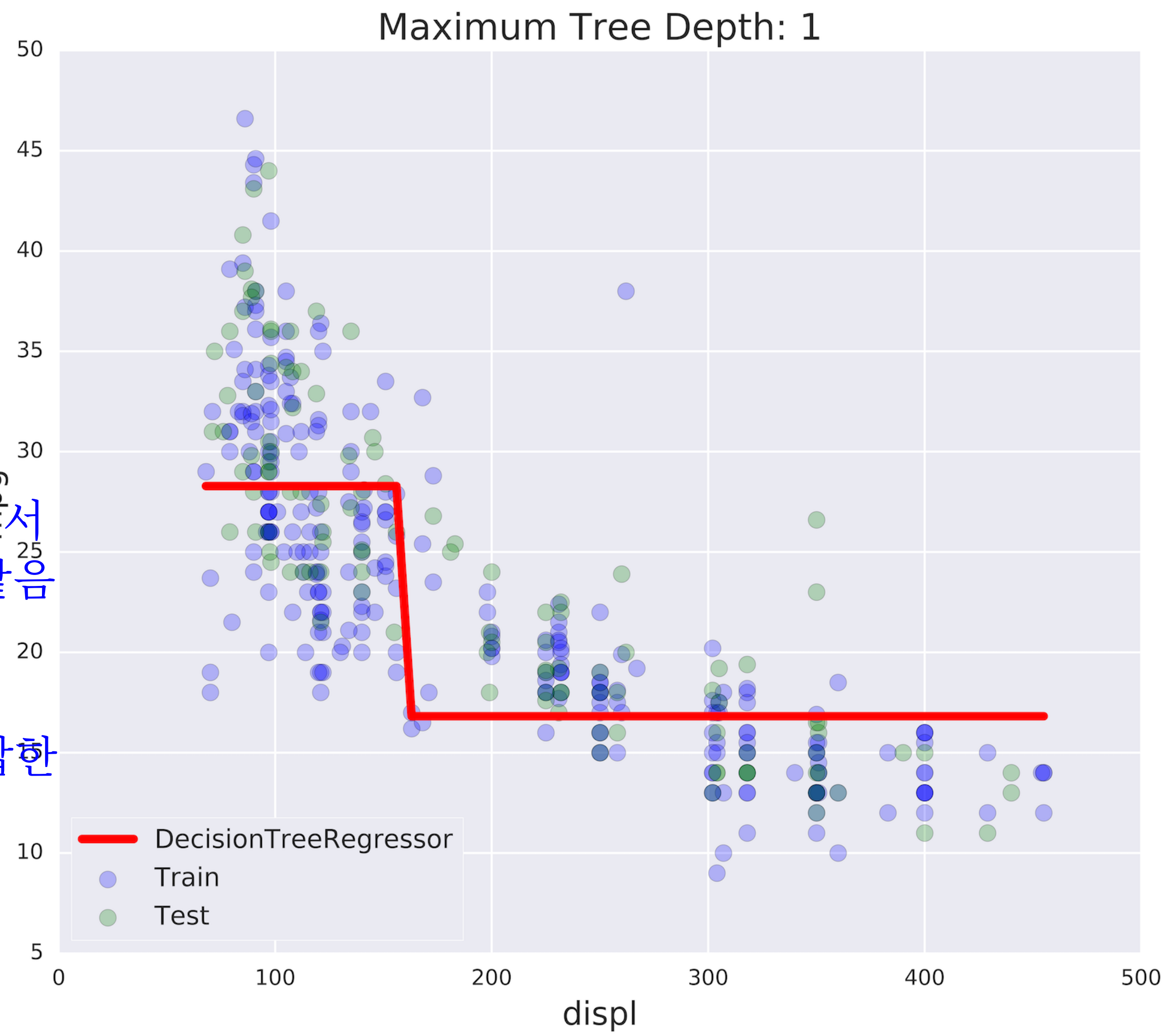
모델이 training set를 overfitting하면
보이지 않는 데이터 세트에 대한 예측력이 매우 낮아짐

모델은 noise있다는 것을 기억하고
낮은 train_set 오류와 높은 test set 오류를 달성



DecisionTree Regression 분석의 예측

Underfitting



예측이 여기 빨간색으로 표시된 회귀 트리에서
training set 오류는 test set 오류와 거의 같음

그러나 두 오류 모두 상대적으로 높음
train 된 model은 feature과 label 간의 복잡한
종속성을 포착 할만큼 유연하지 않음

Generalization Error

- **Generalization Error of $\hat{f}$**: Does $\hat{f}$ generalize well on unseen data?
- It can be decomposed as follows: Generalization Error of $\hat{f} = bias^2 + variance + \text{irreducible error}$

model의 generalization Error는 model이 보이지 않는 데이터에 대해 얼마나 일반화하는지 나타냄
이는 bias, variance, irreducible error 세가지 용어로 나눌 수 있음
(irreducible error(줄일 수 없는 에러) : 우리가 알수없는 오차(오차항)에 대한 분산)

MSE = 분산 + 편파성^2 + irreducible error

분산(variance): 모델이 예측한 데이터의 분포정도, 전체 데이터 집합 중 다른 학습데이터를 사용하였을 때 모델f가 변하는 정도

편파성(bias): 모델이 예측한 데이터와 실제 데이터간의 차이의 정도, 학습 알고리즘에서 잘못된 가정을 했을 때 발생하는 오차

irreducible error: 우리가 알수없는 오차(오차항)에 대한 분산

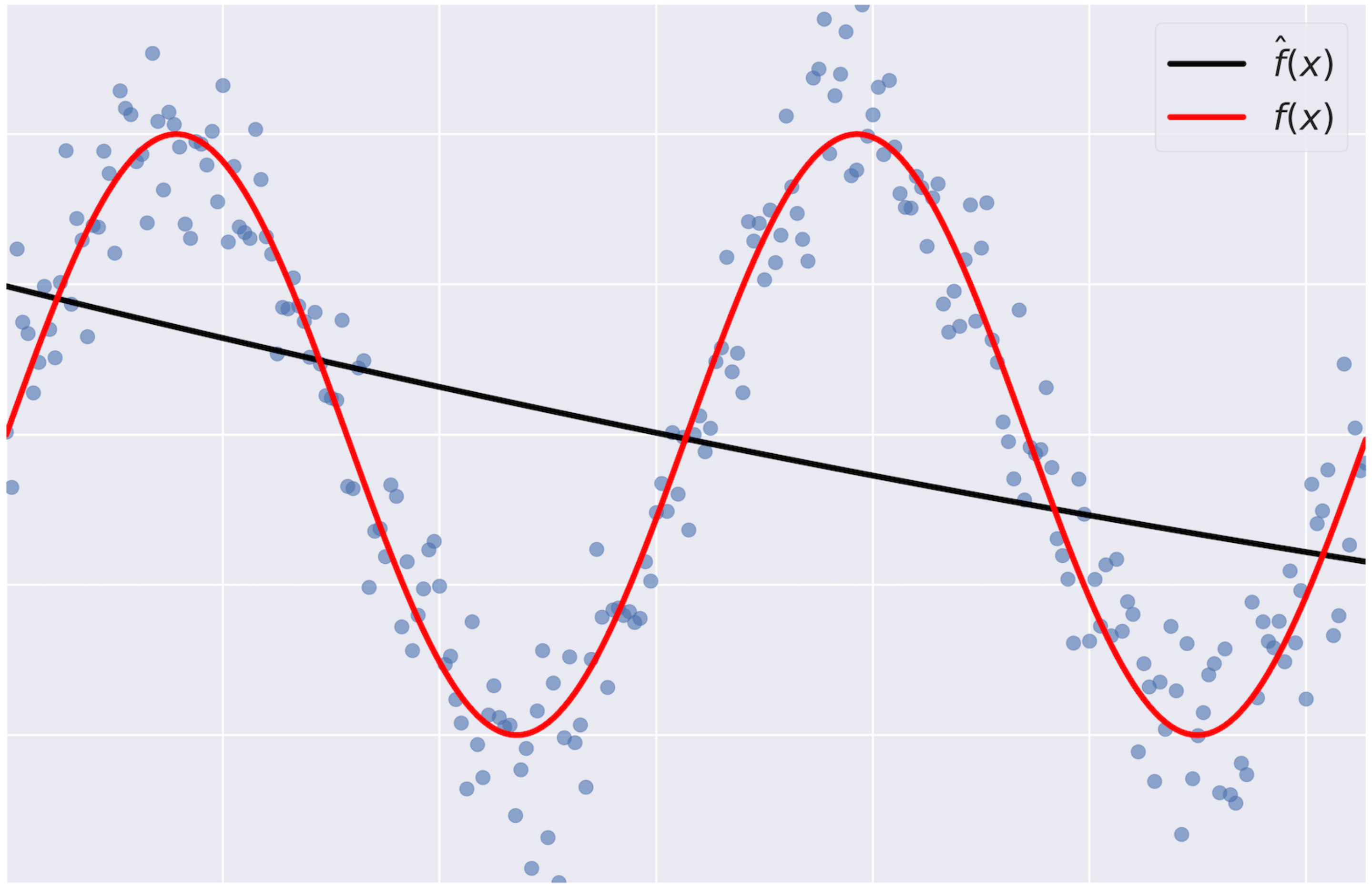
-> 복잡한 모델일수록 분산이 높고, 간단한 모델일수록 편파성이 높다(즉 모델의 복잡도에 따라 분산과 편파성의 정도가 반비례한다.)

-> 즉, 줄일 수 있는 오류를 최대한 줄여야하며, 분산과 편파성의 합이 가장 적은 모델을 설계하도록 해야한다.

Bias

- **Bias:** error term that tells you, on average, how much $\hat{f} \neq f$.

평균적으로 fhat과 f가 얼마나 다른지 알려줌



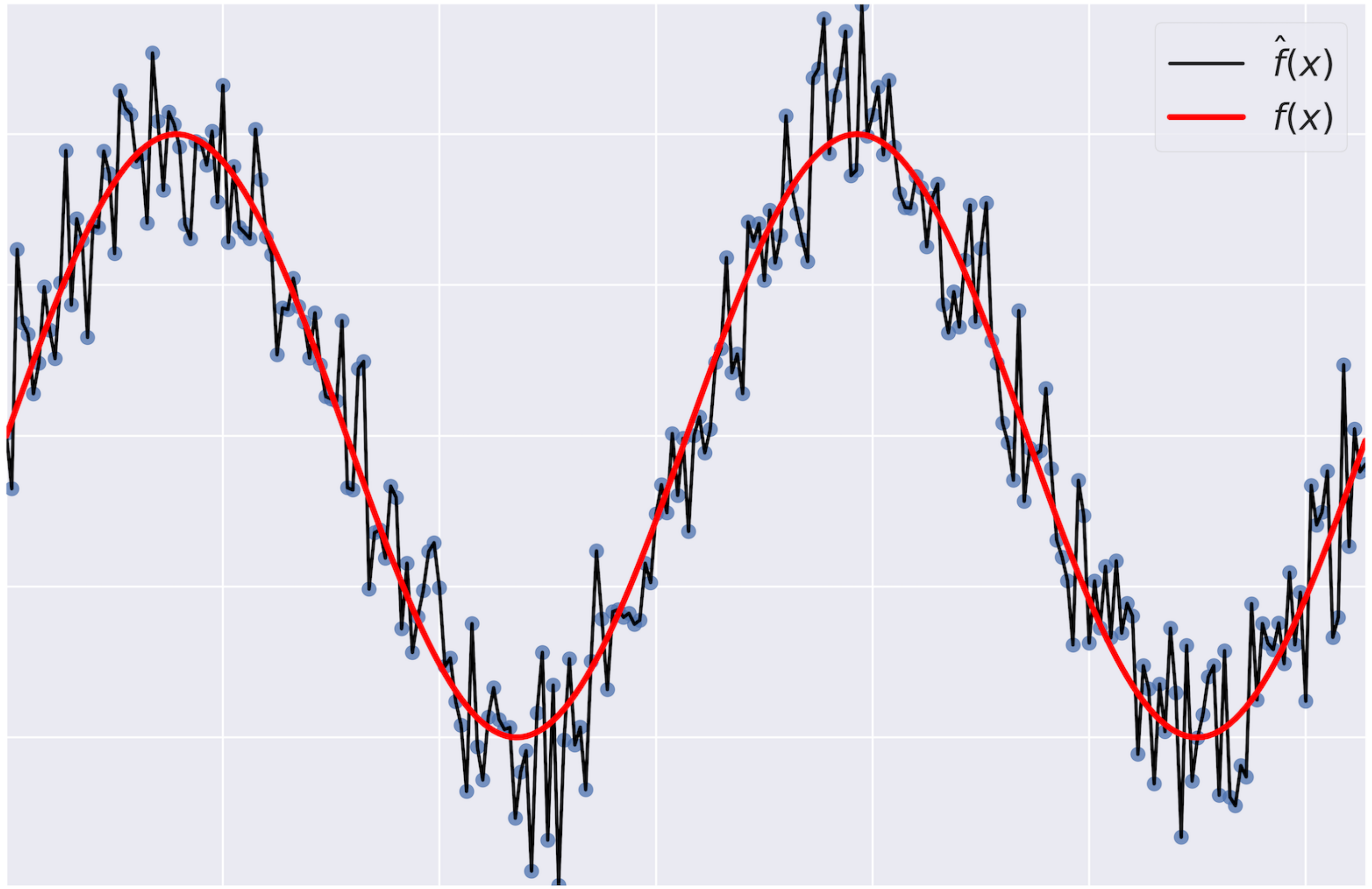
f_hat은 f에 근접할만큼 유연하지 않음

높은 bias model은 underfitting으로 이어짐

Variance

- **Variance:** tells you how much $\hat{f}$ is inconsistent over different training sets.

다른 training set에서 얼마나 많은 fhat이 일치하지 않는지 알렸무



fhat은 training data point를 너무 가깝게 따라가기 때문에 빨간색으로 표시된 함수 f를 놓침

variance(분산) model이 크면 underfitting 발생

Model Complexity

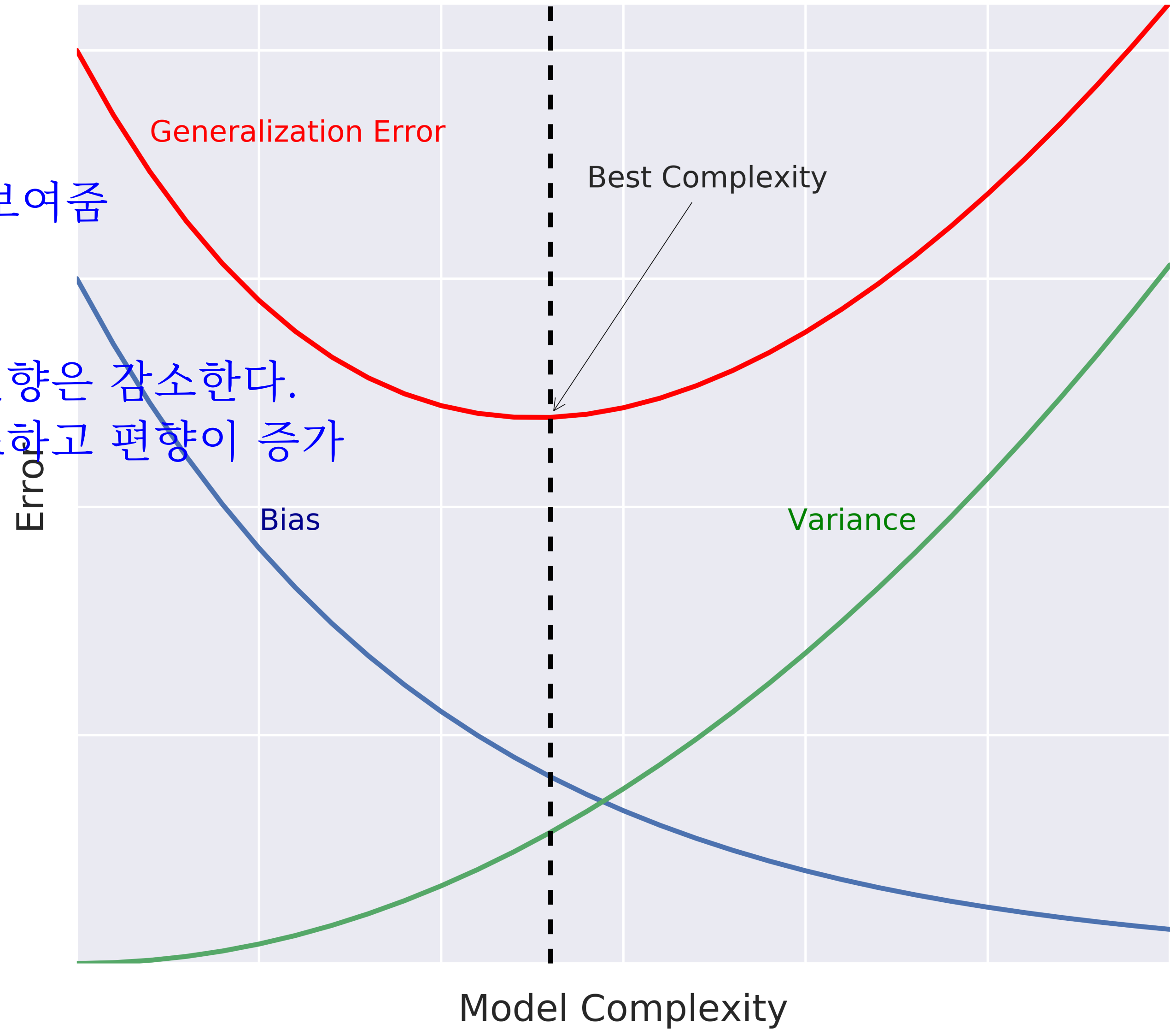
- **Model Complexity:** sets the flexibility of $\hat{f}$. nodel의 복잡성은 실제 함수 f에 근접하도록 유연성을 설정
- Example: Maximum tree depth, Minimum samples per leaf, ... ex) 최대 트리 깊이를 늘리면 decision tree의 복잡성이 증가

Bias-Variance Tradeoff

목표는가장 낮은 일반화 오류를 달성하는 model 복잡성을 찾는 것

여기의 diagram은 최상의 model 복잡성이
가장 낮은 일반화 오류에 해당하는 방식을 보여줌

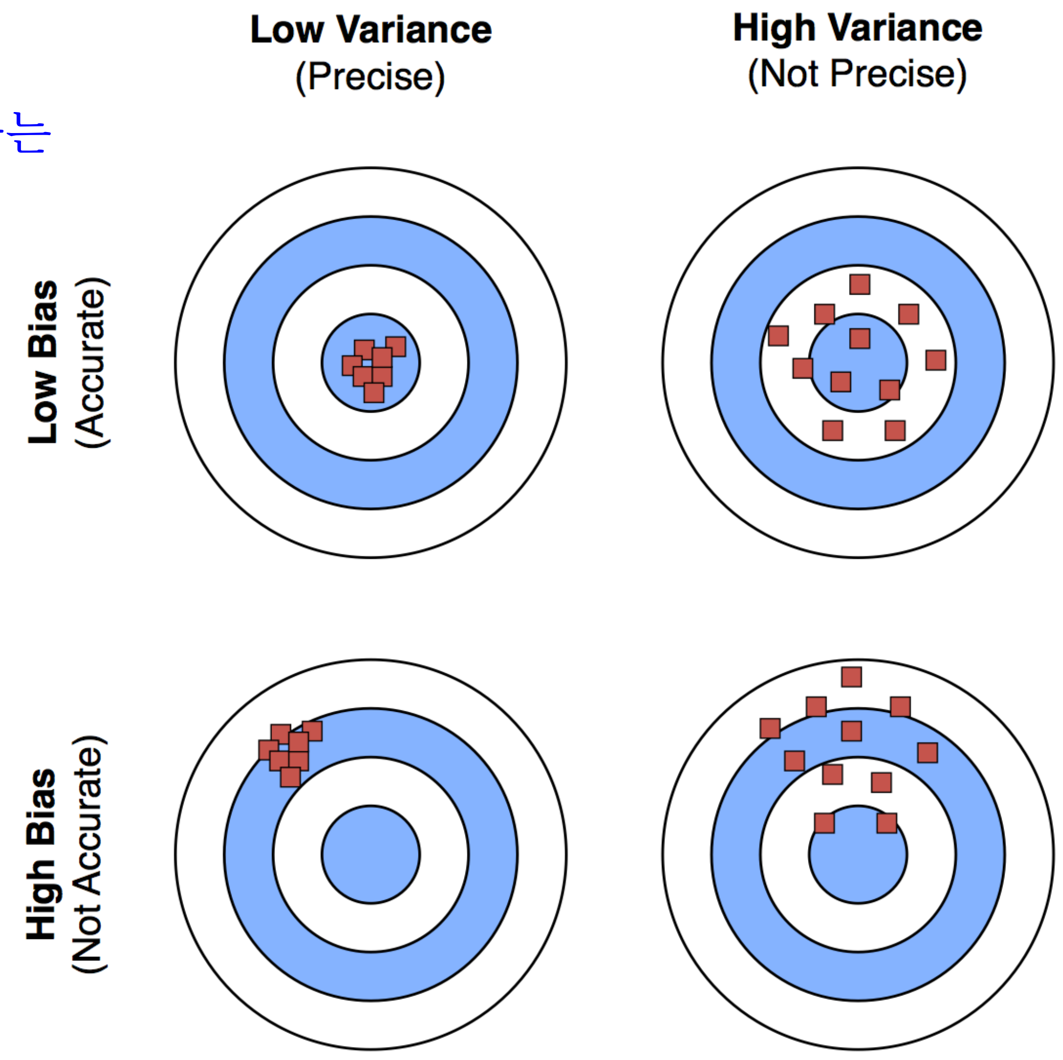
모델 복잡성이 증가하면 분산은 증가하고 편향은 감소한다.
반대로 모델 복잡성이 감소하면 분산이 감소하고 편향이 증가
=> 편향과 분산 사이의 균형을 찾아야함
이를 bias - variance tradeoff라고 함



Bias-Variance Tradeoff: A Visual Explanation

fhat의 중심이 f인 사격 목표의 중심을 목표로 하는
근사 fhat

fhat이 낮은 bias과 낮은 variance이면
샷이 중심 주위에 가깝게 군집화됨
만약 fhat이 높은 variacne과 높은 bias라면,
사격 목표 주위에 퍼짐



Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Diagnosing Bias and Variance Problems

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

편향 및 분산 문제를 진단하는 방법을 배

Elie Kawerk
Data Scientist

Estimating the Generalization Error

- How do we estimate the generalization error of a model? fhat이라는 label이 붙은 fhat의 일반화 오류를 어떻게 추정할까?
- Cannot be done directly because:
  - f is unknown,
  - usually you only have one dataset,
  - noise is unpredictable.

Estimating the Generalization Error

Solution:

- split the data to training and test sets,
- fit $\hat{f}$ to the training set,
- evaluate the error of $\hat{f}$ on the **unseen** test set.
- generalization error of $\hat{f} \approx$ test set error of $\hat{f}$. fhat의 일반화 오류는 대략 test set에 대한 fhat의 오류로 근사

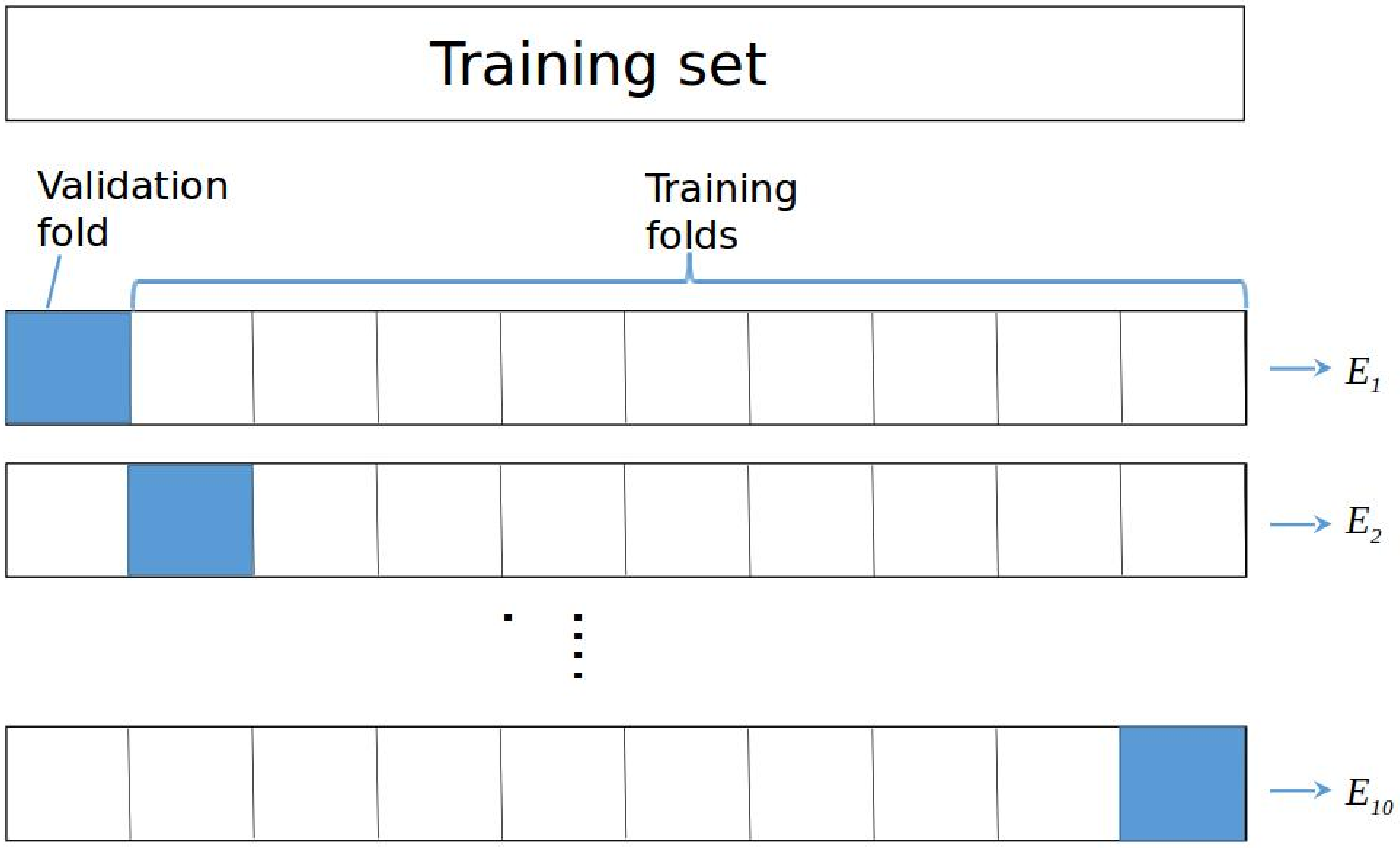
Better Model Evaluation with Cross-Validation

- Test set should not be touched until we are confident about $\hat{f}$'s performance.
- Evaluating $\hat{f}$ on training set: biased estimate, $\hat{f}$ has already seen all training points.
- Solution $\rightarrow$ Cross-Validation (CV):
  - K-Fold CV,
  - Hold-Out CV.

fhat의 최종 성능 또는 오류를 평가하는 데만 사용
fhat이 적합할 때 이미 train set에 노출되었기 때문에 train set에 대한 fhat의 성능을 평가하면 오류에 대한 낙관적인 추정을 생성할 수 있음
fhat의 성능에 대한 신뢰할 수 있는 추정치를 얻으려면 교차 검증 또는 CV라는 기법을 사용해야함.
CV는 K - Fold - CV 또는 hold - out - cv를 사용하여 수행할 수 있음

K-Fold CV

10



K-Fold CV

$$\text{CV error} = \frac{E_1 + \ldots + E_{10}}{10}$$

CV - error는 10개의 얻은 오류의 평균으로 계산

Diagnose Variance Problems

- If $\hat{f}$ suffers from **high variance**: CV error of $\hat{f}$ > training set error of $\hat{f}$.
  fhat의 교차 검증 오류를 계산 한 후에는 fhat의 훈련 세트 오류보다 큰지 확인할 수 있음
- $\hat{f}$ is said to overfit the training set. To remedy overfitting: 더 크면 fhat은 높은 variance로 고통받음
  => 이경우 fhat은 train_set에 overfitting함
  - decrease model complexity, 이를 해결하려면 fhat의 복잡성을 줄여야함
  - for ex: decrease max depth, increase min samples per leaf, ...
  - gather more data, ..

Diagnose Bias Problems

교차 검증 오류는 training set error와 거의 같지만 원하는 오류보다 훨씬 큼

- if $\hat{f}$ suffers from high bias: CV error of $\hat{f}$ $\approx$ training set error of $\hat{f}$ $>>$ desired error.
- $\hat{f}$ is said to underfit the training set. To remedy underfitting:
  - increase model complexity (해결하기 위해) model의 복잡성을 높이거나 문제와 관련된 더 많은 feature를 수집
  - for ex: increase max depth, decrease min samples per leaf, ...
  - gather more relevant features

K-Fold CV in sklearn on the Auto Dataset

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import cross_val_score
# Set seed for reproducibility
SEED = 123
# 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,
                                                    random_state=SEED)
# Instantiate decision tree regressor and assign it to 'dt'
dt = DecisionTreeRegressor(max_depth=4,
                           min_samples_leaf=0.14,
                           random_state=SEED)
```

K-Fold CV in sklearn on the Auto Dataset

```
# Evaluate the list of MSE ontained by 10-fold CV
# Set n_jobs to -1 in order to exploit all CPU cores in computation
MSE_CV = - cross_val_score(dt, X_train, y_train, cv= 10,
                           scoring='neg_mean_squared_error',
                           n_jobs = -1)

# Fit 'dt' to the training set
dt.fit(X_train, y_train)
# Predict the labels of training set
y_predict_train = dt.predict(X_train)
# Predict the labels of test set
y_predict_test = dt.predict(X_test)
```

cross_val_score()는 평균 제곱 오차를 직접 계산할 수 없기 때문에 scoring 매개 변수 설정

음의 평균 제곱 오차를 계산하기 위해 10배 교차 검증 및 neg_mean_squareed_error에 점수 매기기

n_jobs를 - 1로 설정하여 계산에서 사용 가능한 모든 CPU 활용
결과 10배에서 달성된 10개의 음의 평균 제곱 오차의 숫자 배열이고, 결과에 - 1을 곱하면 CV - MSE 배열을 얻을 수 있음

```
# CV MSE
print('CV MSE: {:.2f}'.format(MSE_CV.mean()))
```

```
CV MSE: 20.51
```

```
# Training set MSE
print('Train MSE: {:.2f}'.format(MSE(y_train, y_predict_train)))
```

training set error가 CV error보다 작으므로 dt가 train set에 overfitting하고 높은 variance로 고통받고 있음

```
Train MSE: 15.30
```

```
# Test set MSE
print('Test MSE: {:.2f}'.format(MSE(y_test, y_predict_test)))
```

```
Test MSE: 20.92
```

CV 및 test set error가 대략적으로 동일한지 확인

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Ensemble Learning

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

Advantages of CARTs

- Simple to understand.
- Simple to interpret.
- Easy to use.
- Flexibility: ability to describe non-linear dependencies.
- Preprocessing: no need to standardize or normalize features, ...

Limitations of CARTs

- Classification: can only produce orthogonal decision boundaries.
- Sensitive to small variations in the training set. 때로 train set에서 단일 point가 제거되면 CART의 학습 된 매개 변수가 크게 변경될 수 있음
- High variance: unconstrained CARTs may overfit the training set. 이러한 경우 train set에 과적합 할 수 있음
- Solution: ensemble learning.

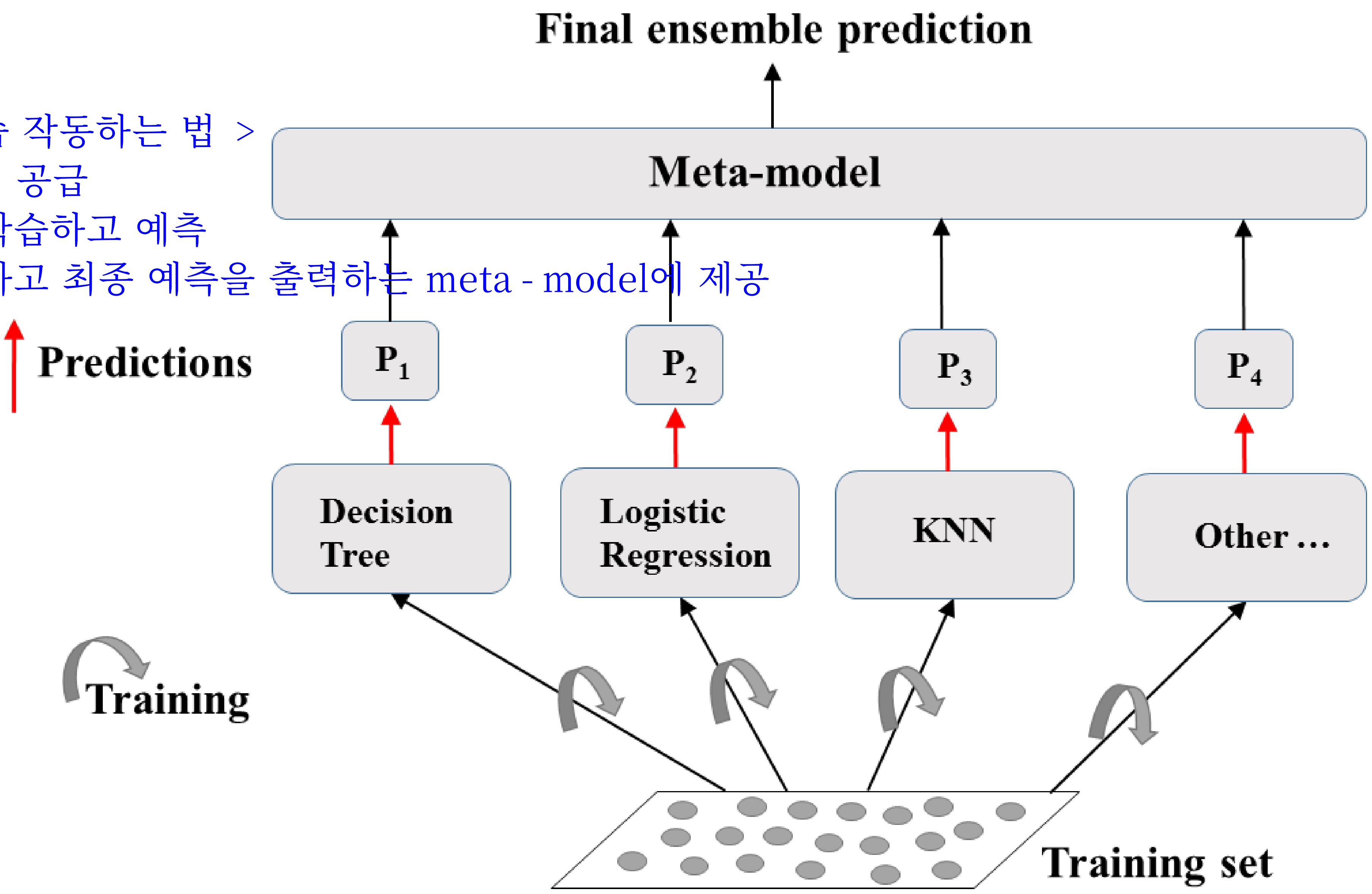
Ensemble Learning

- Train different models on the same dataset.
- Let each model make its predictions.
- Meta-model: aggregates predictions of individual models. meta - model은 개별 model의 예측을 집계하고 최종 예측을 출력
- Final prediction: more robust and less prone to errors. 최종 예측은 각 개별 모델보다 더 강력하고 오류가 적음
- Best results: models are skillful in different ways.

(일부 모델이 잘못된 예측을하는 경우 다른 모델은 이러한 오류를 보상해야함)
이 경우 meta - model의 예측이 더 강력함

Ensemble Learning: A Visual Explanation

< 분류 문제에서 앙상블 학습 작동하는 법 >
1. train set는 다른 분류기에 공급
2. 각분류기는 매개 변수를 학습하고 예측
3. 이러한 예측은 이를 집계하고 최종 예측을 출력하는 meta - model에 제공

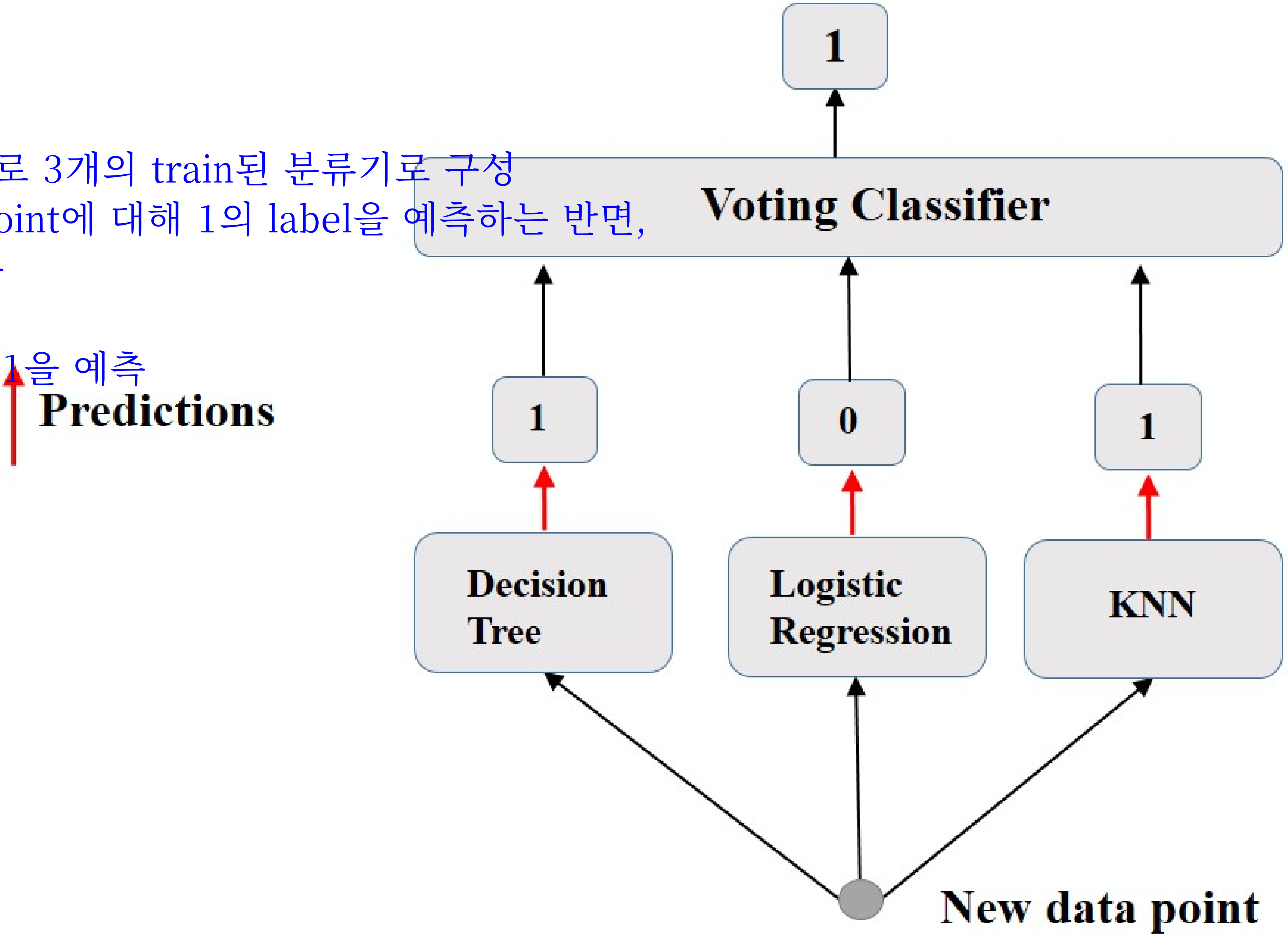


Ensemble Learning in Practice: Voting Classifier

- Binary classification task.
- N classifiers make predictions: P_1, P_2, ..., P_N with P_i = 0 or 1.
- Meta-model prediction: hard voting. meta - model은 hard voting을 통해 최종 예측을 출력

Hard Voting

여기 diagram에 표시된대로 3개의 train된 분류기로 구성
분류자 1과 3은 새 data point에 대해 1의 label을 예측하는 반면,
분류자 2는 label 0을 예측
=> 1은 2표, 2는 1표
=> 결론적으로 분류기는 1을 예측



Voting Classifier in sklearn (Breast-Cancer dataset)

```
# Import functions to compute accuracy and split data
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Import models, including VotingClassifier meta-model
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.ensemble import VotingClassifier

# Set seed for reproducibility
SEED = 1
```

Voting Classifier in sklearn (Breast-Cancer dataset)

```
# 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,
                                                    random_state= SEED)
# Instantiate individual classifiers
lr = LogisticRegression(random_state=SEED)
knn = KNN()
dt = DecisionTreeClassifier(random_state=SEED)
# Define a list called classifier that contains the tuples (classifier_name, classifier)
classifiers = [('Logistic Regression', lr),
               ('K Nearest Neighbours', knn),
               ('Classification Tree', dt)]
```

```python
# Iterate over the defined list of tuples containing the classifiers
for clf_name, clf in classifiers:
    #fit clf to the training set
    clf.fit(X_train, y_train)

    # Predict the labels of the test set
    y_pred = clf.predict(X_test)

    # Evaluate the accuracy of clf on the test set
    print('{:s} : {:.3f}'.format(clf_name, accuracy_score(y_test, y_pred)))
```

```
Logistic Regression: 0.947
K Nearest Neighbours: 0.930
Classification Tree: 0.930
```

Voting Classifier in sklearn (Breast-Cancer dataset)

```
# Instantiate a VotingClassifier 'vc'
vc = VotingClassifier(estimators=classifiers)

# Fit 'vc' to the traing set and predict test set labels
vc.fit(X_train, y_train)
y_pred = vc.predict(X_test)

# Evaluate the test-set accuracy of 'vc'
print('Voting Classifier: {.3f}'.format(accuracy_score(y_test, y_pred)))
```

```
Voting Classifier: 0.953
```

=> 정확도가 앙상블의 개별 모델이 달성한 것보다 높음

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON