AdaBoost

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

Boosting

- **Boosting**: Ensemble method combining several weak learners to form a strong learner.
- **Weak learner**: Model doing slightly better than random guessing.
- Example of weak learner: Decision stump (CART whose maximum depth is 1).

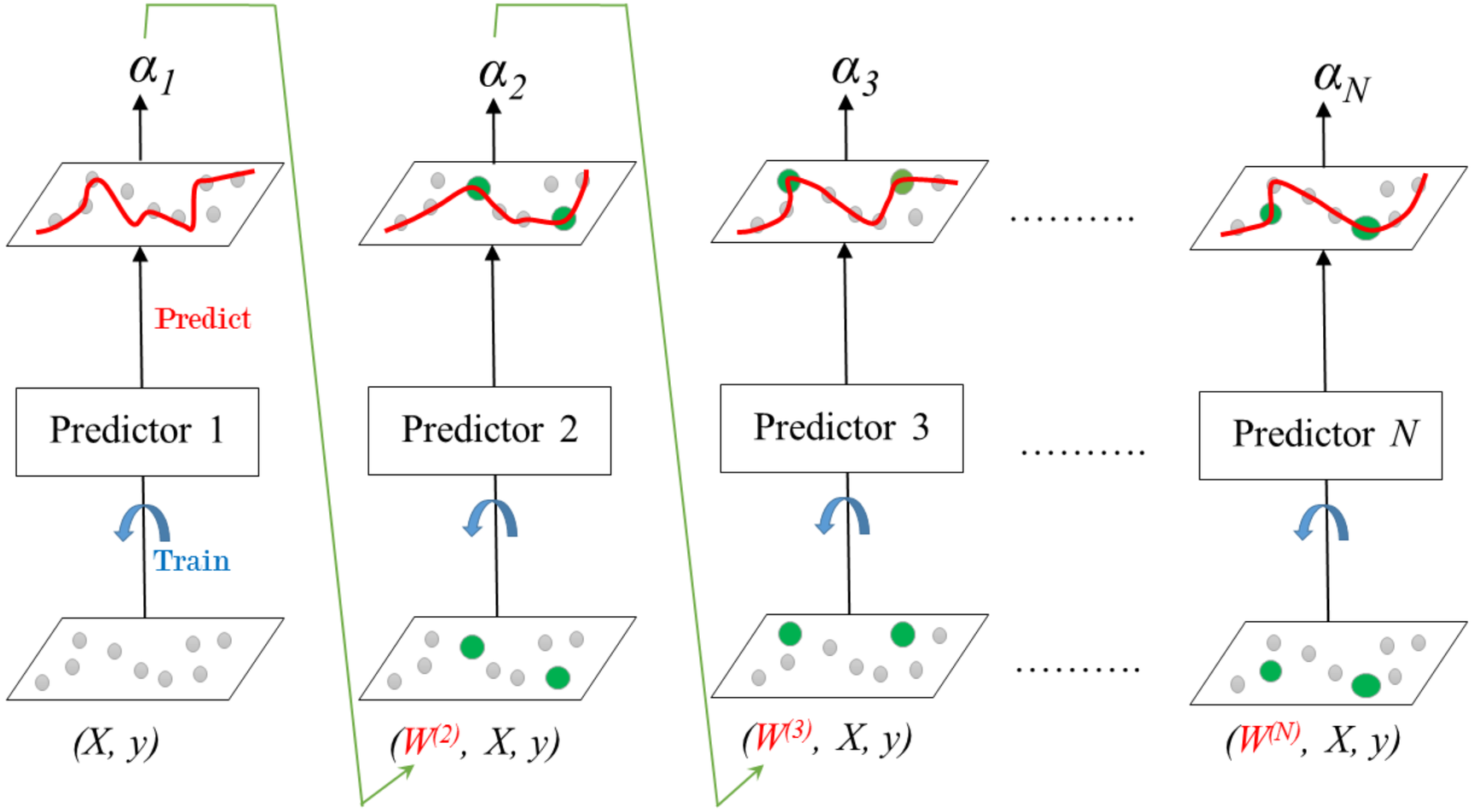
Boosting

- Train an ensemble of predictors sequentially.
- Each predictor tries to correct its predecessor.
- Most popular boosting methods:
  - AdaBoost,
  - Gradient Boosting.

Adaboost

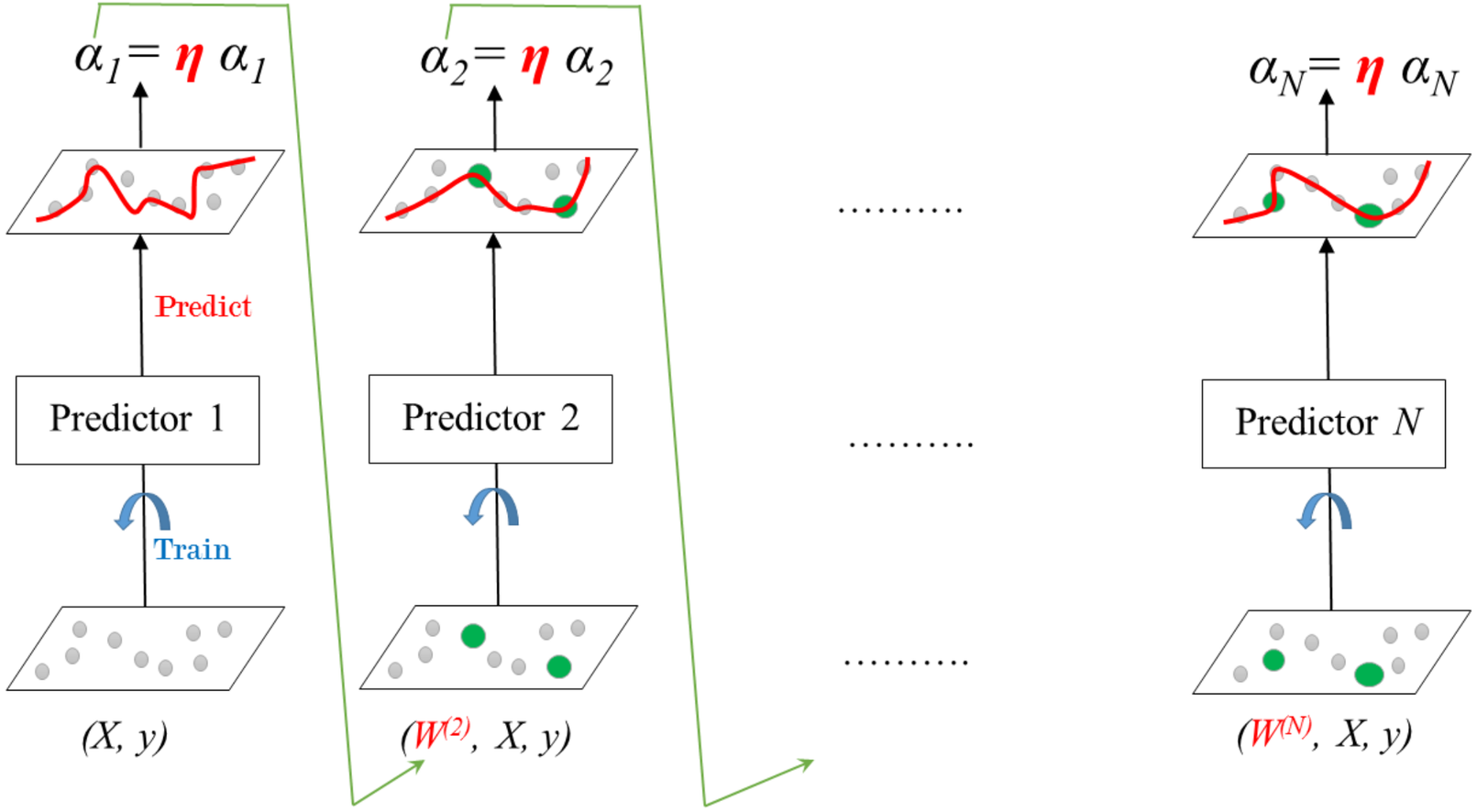
- Stands for **Ada**ptive **Boost**ing.
- Each predictor pays more attention to the instances wrongly predicted by its predecessor.
- Achieved by changing the weights of training instances.
- Each predictor is assigned a coefficient α.
- α depends on the predictor's training error.

AdaBoost: Training



Learning Rate

Learning rate: $0 < \eta \leq 1$



AdaBoost: Prediction

- Classification:
  - Weighted majority voting.
  - In sklearn: `AdaBoostClassifier`.
- Regression:
  - Weighted average.
  - In sklearn: `AdaBoostRegressor`.

AdaBoost Classification in sklearn (Breast Cancer dataset)

```
# Import models and utility functions
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_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=1, random_state=SEED)

# Instantiate an AdaBoost classifier 'adab_clf'
adb_clf = AdaBoostClassifier(base_estimator=dt, n_estimators=100)



# Fit 'adb_clf' to the training set
adb_clf.fit(X_train, y_train)

# Predict the test set probabilities of positive class
y_pred_proba = adb_clf.predict_proba(X_test)[:,1]



# Evaluate test-set roc_auc_score
adb_clf_roc_auc_score = roc_auc_score(y_test, y_pred_proba)
```

AdaBoost Classification in sklearn (Breast Cancer dataset)

```
# Print adb_clf_roc_auc_score
print('ROC AUC score: {:.2f}'.format(adb_clf_roc_auc_score))
```

```
ROC AUC score: 0.99
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Gradient Boosting (GB)

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

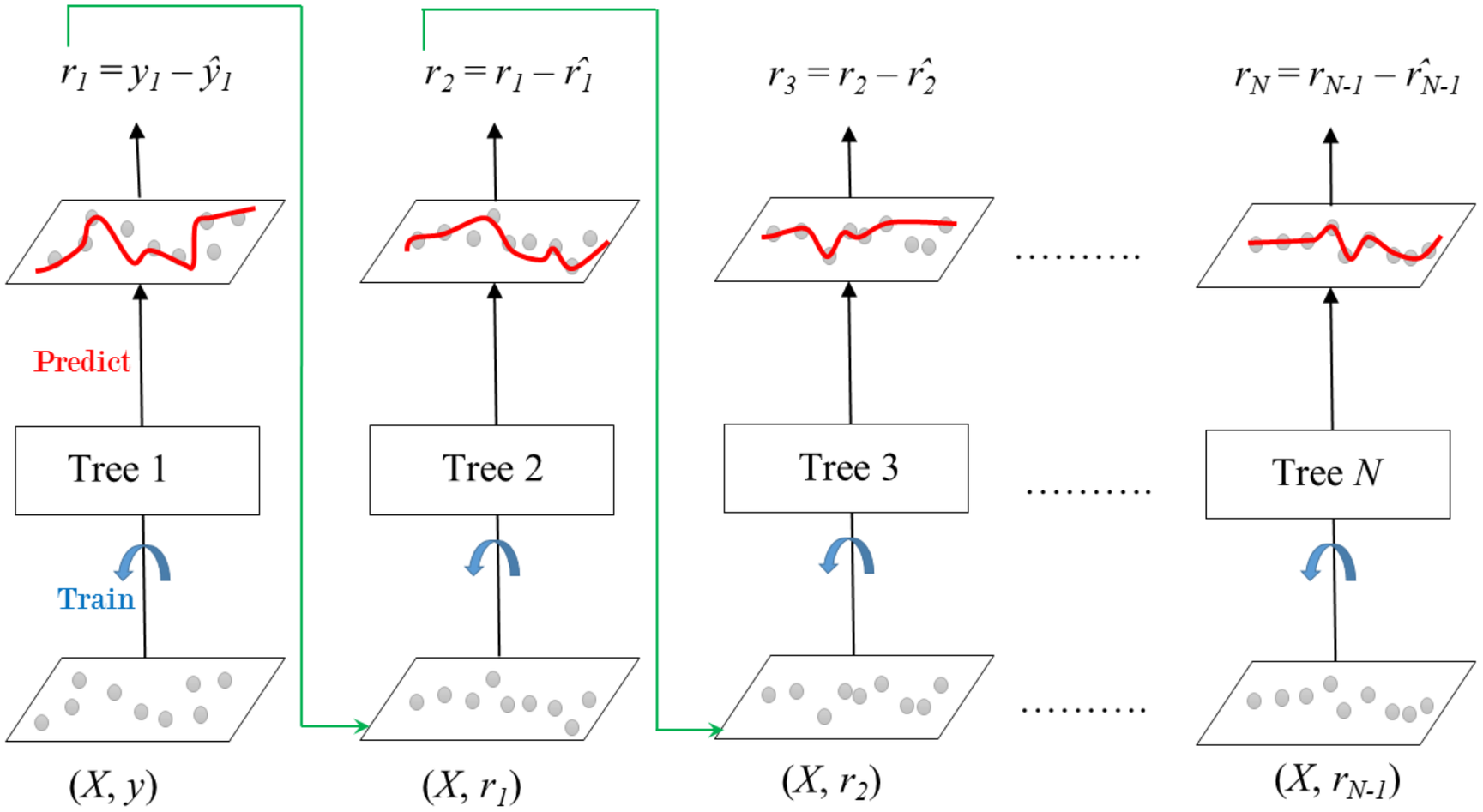
Elie Kawerk
Data Scientist

datacamp

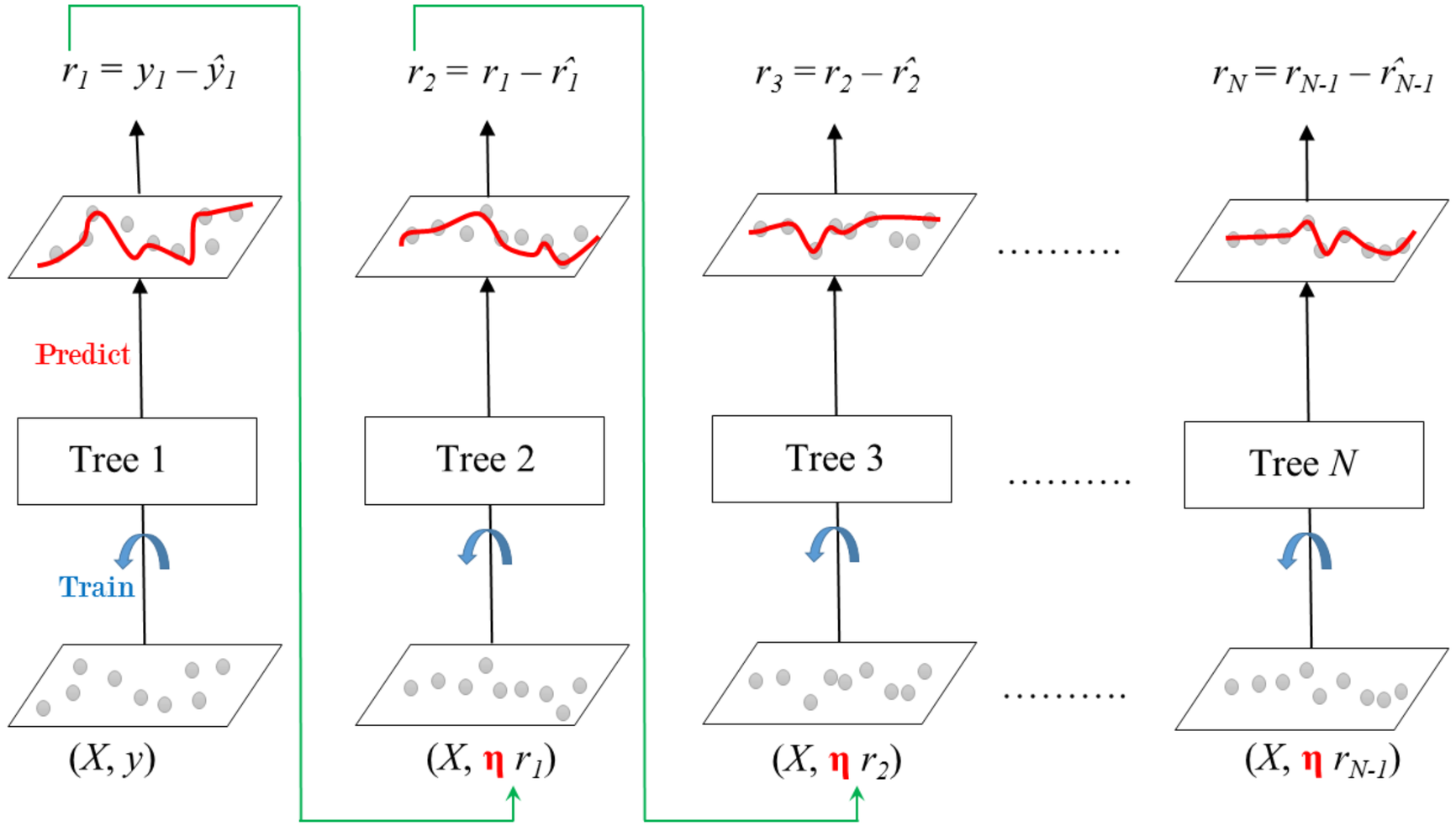
Gradient Boosted Trees

- Sequential correction of predecessor's errors.
- Does not tweak the weights of training instances.
- Fit each predictor is trained using its predecessor's residual errors as labels.
- Gradient Boosted Trees: a CART is used as a base learner.

Gradient Boosted Trees for Regression: Training



Shrinkage


$r_1 = y_1 - \hat{y}_1$
$r_2 = r_1 - \hat{r_1}$
$r_3 = r_2 - \hat{r_2}$
$r_N = r_{N-1} - \hat{r}_{N-1}$
Predict
Tree 1
Tree 2
Tree 3
Tree N
Train
(X, y)
$(X, \eta\ r_1)$
$(X, \eta\ r_2)$
$(X, \eta\ r_{N-1})$

Gradient Boosted Trees: Prediction

- Regression:
  - $y_{pred} = y_1 + \eta r_1 + ... + \eta r_N$
  - In sklearn: `GradientBoostingRegressor`.
- Classification:
  - In sklearn: `GradientBoostingClassifier`.

Gradient Boosting in sklearn (auto dataset)

```
# Import models and utility functions
from sklearn.ensemble import GradientBoostingRegressor
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 GradientBoostingRegressor 'gbt'
gbt = GradientBoostingRegressor(n_estimators=300, max_depth=1, random_state=SEED)

# Fit 'gbt' to the training set
gbt.fit(X_train, y_train)

# Predict the test set labels
y_pred = gbt.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: {:.2f}'.format(rmse_test))
```

```
Test set RMSE: 4.01
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Stochastic Gradient Boosting (SGB)

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON

Elie Kawerk
Data Scientist

datacamp

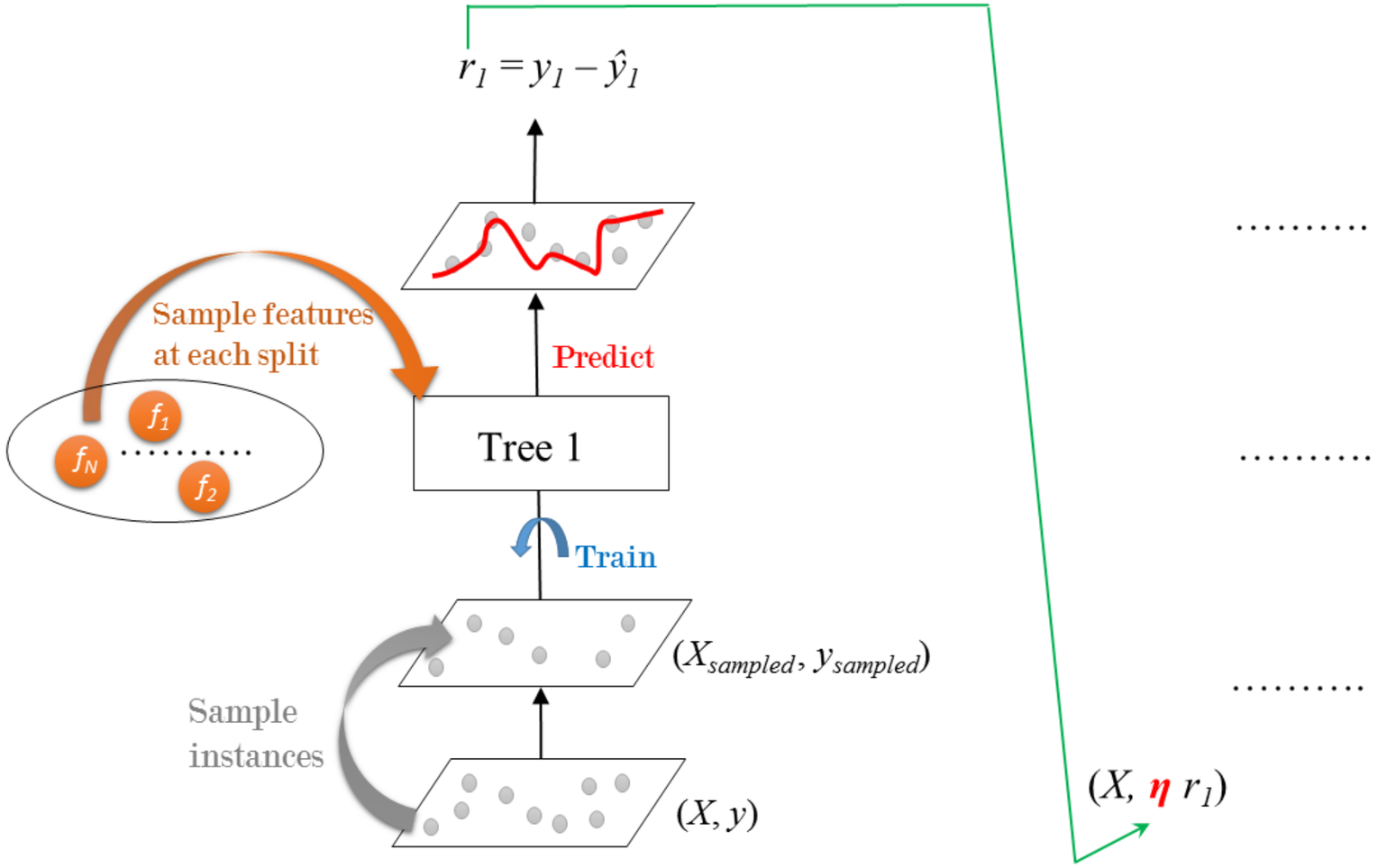
Gradient Boosting: Cons

- GB involves an exhaustive search procedure.
- Each CART is trained to find the best split points and features.
- May lead to CARTs using the same split points and maybe the same features.

Stochastic Gradient Boosting

- Each tree is trained on a random subset of rows of the training data.
- The sampled instances (40%-80% of the training set) are sampled without replacement.
- Features are sampled (without replacement) when choosing split points.
- Result: further ensemble diversity.
- Effect: adding further variance to the ensemble of trees.

Stochastic Gradient Boosting: Training



Stochastic Gradient Boosting in sklearn (auto dataset)

```
# Import models and utility functions
from sklearn.ensemble import GradientBoostingRegressor
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)
```

Stochastic Gradient Boosting in sklearn (auto dataset)

```
# Instantiate a stochastic GradientBoostingRegressor 'sgbt'
sgbt = GradientBoostingRegressor(max_depth=1,
                                 subsample=0.8,
                                 max_features=0.2,
                                 n_estimators=300,
                                 random_state=SEED)

# Fit 'sgbt' to the training set
sgbt.fit(X_train, y_train)

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

Stochastic Gradient Boosting in sklearn (auto dataset)

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

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

```
Test set RMSE: 3.95
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN PYTHON