

# The strength of "weak" models

ENSEMBLE METHODS IN PYTHON



**Román de las Heras**

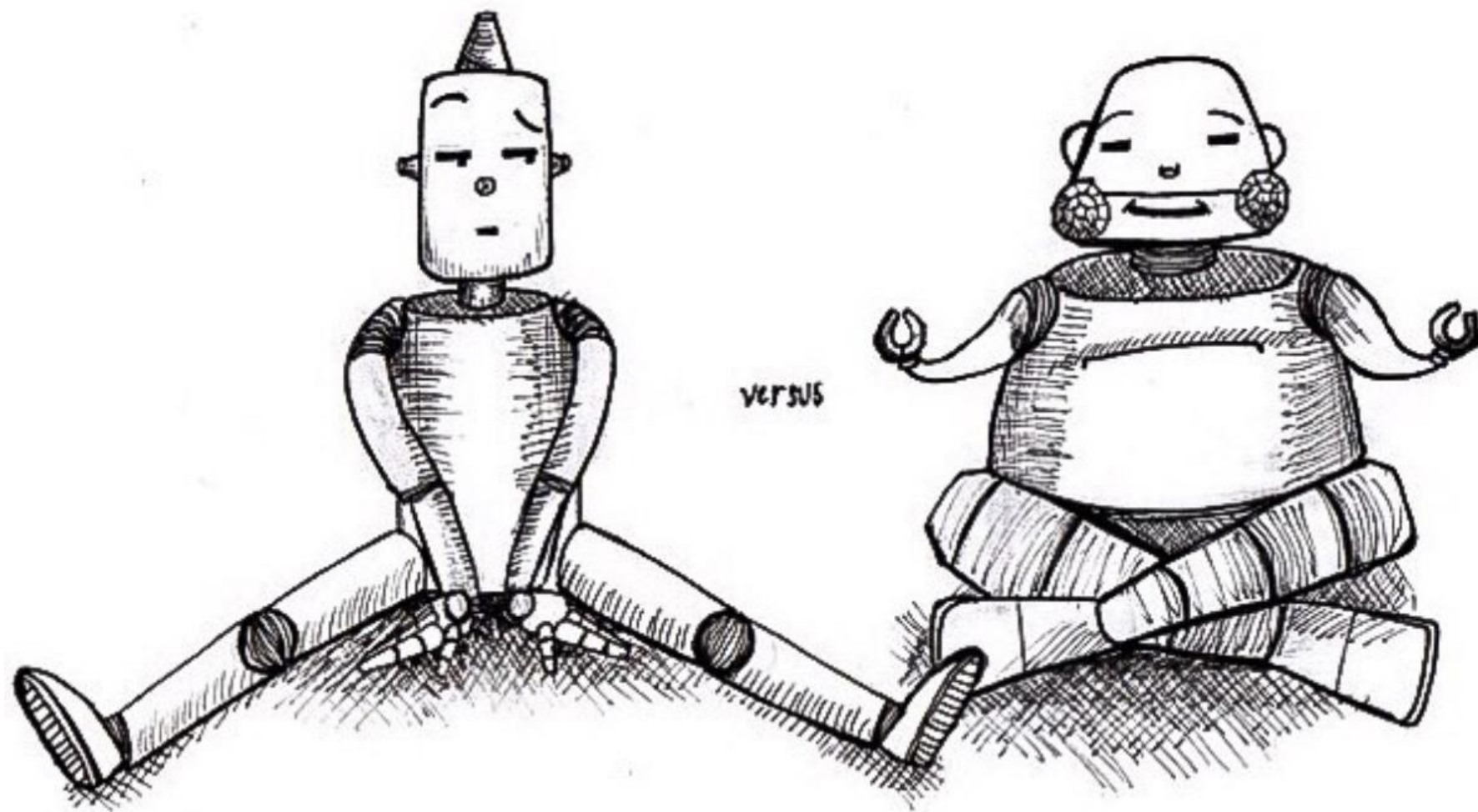
Senior Data Scientist, Chartboost

# "Weak" model

## Voting and Averaging:

- Small number of estimators
- **Fine-tuned** estimators
- Individually trained

New concept: "*weak*" estimator



**"Weak" model**

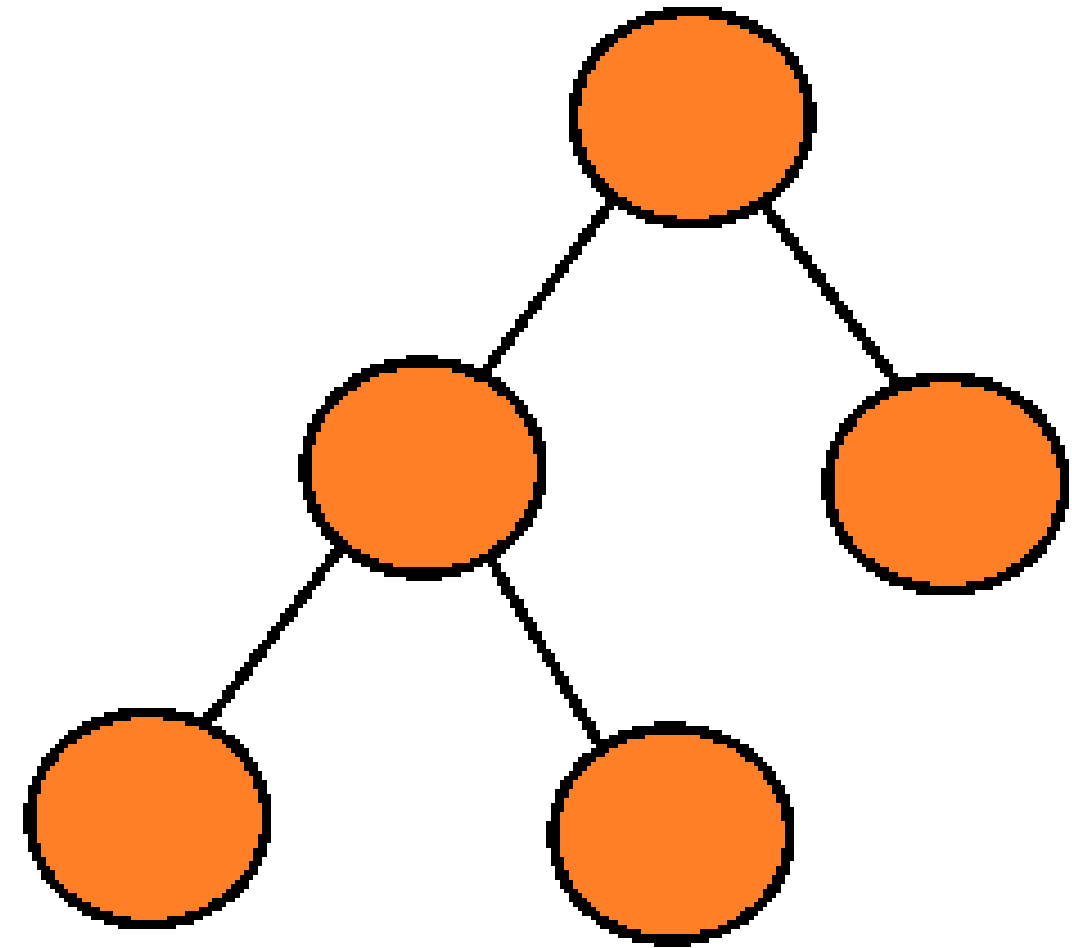
**Fine-tuned model**

# Properties of "weak" models

## Weak estimator

- Performance better than random guessing
- Light model
- Low training and evaluation time

## Example: Decision Tree



# Examples of "weak" models

Some "weak" models:

- Decision tree: small depth
- Logistic Regression
- Linear Regression
- Other restricted models

Sample code:

```
model = DecisionTreeClassifier(  
    max_depth=3  
)  
model = LogisticRegression(  
    max_iter=50, C=100.0  
)  
model = LinearRegression()
```

**Let's practice!**  
ENSEMBLE METHODS IN PYTHON

# Bootstrap aggregating

ENSEMBLE METHODS IN PYTHON



**Román de las Heras**

Senior Data Scientist, Chartboost

# Heterogeneous vs Homogeneous Ensembles

## Heterogeneous:

- Different algorithms (fine-tuned)
- Small amount of estimators
- Voting, Averaging, and Stacking

## Homogeneous:

- The same algorithm ("weak" model)
- Large amount of estimators
- Bagging and Boosting



# Condorcet's Jury Theorem

## Requirements:

- Models are independent
- Each model performs better than random guessing
- All individual models have similar performance

**Conclusion:** Adding more models improves the performance of the ensemble (*Voting* or *Averaging*), and this approaches 1 (100%)



*Marquis de Condorcet, French philosopher and mathematician*

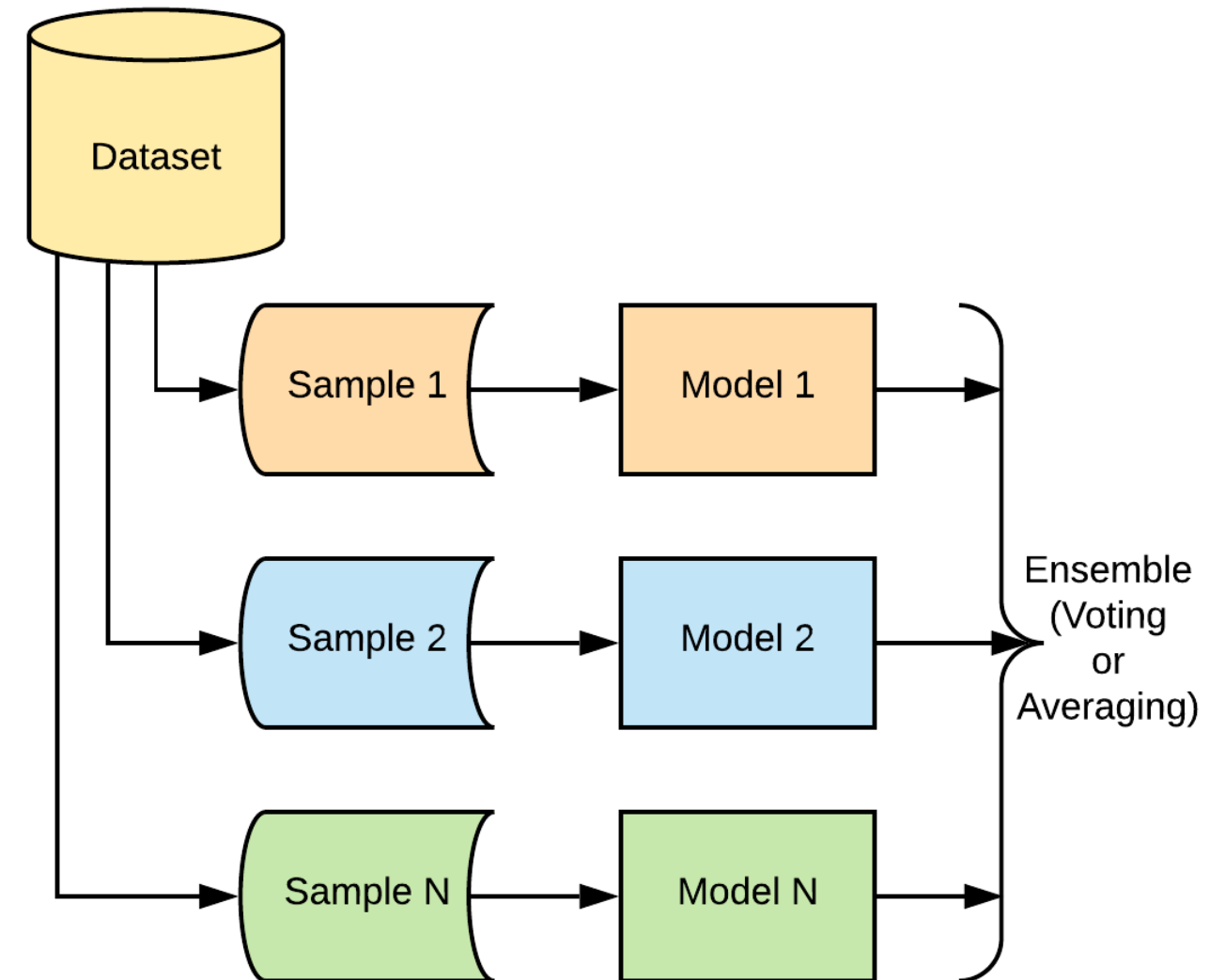
# Bootstrapping

Bootstrapping requires:

- Random subsamples
- Using replacement

Bootstrapping guarantees:

- **Diverse crowd:** different datasets
- **Independent:** separately sampled



# Pros and cons of bagging

## Pros

- Bagging usually reduces variance
- Overfitting can be avoided by the ensemble itself
- More stability and robustness

## Cons

- It is computationally expensive

# It's time to practice!

ENSEMBLE METHODS IN PYTHON

# BaggingClassifier: nuts and bolts

ENSEMBLE METHODS IN PYTHON



**Román de las Heras**

Senior Data Scientist, Chartboost

# Heterogeneous vs Homogeneous Functions

## Heterogeneous Ensemble Function

```
het_est = HeterogeneousEnsemble(  
    estimators=[('est1', est1), ('est2', est2), ...],  
    # additional parameters  
)
```

## Homogeneous Ensemble Function

```
hom_est = HomogeneousEnsemble(  
    base_estimator=est_base,  
    n_estimators=chosen_number,  
    # additional parameters  
)
```

# BaggingClassifier

## Bagging Classifier example:

```
# Instantiate the base estimator ("weak" model)
clf_dt = DecisionTreeClassifier(max_depth=3)
```

```
# Build the Bagging classifier with 5 estimators
clf_bag = BaggingClassifier(
    base_estimator=clf_dt,
    n_estimators=5
)
```

```
# Fit the Bagging model to the training set
clf_bag.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = clf_bag.predict(X_test)
```

# BaggingRegressor

## Bagging Regressor example:

```
# Instantiate the base estimator ("weak" model)
reg_lr = LinearRegression()
```

```
# Build the Bagging regressor with 10 estimators
reg_bag = BaggingRegressor(
    base_estimator=reg_lr
)
```

```
# Fit the Bagging model to the training set
reg_bag.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = reg_bag.predict(X_test)
```



# Out-of-bag score

- Calculate the individual predictions using all estimators for which an instance was out of the sample
- Combine the individual predictions
- Evaluate the metric on those predictions:
  - **Classification:** accuracy
  - **Regression:**  $R^2$

```
clf_bag = BaggingClassifier(  
    base_estimator=clf_dt,  
    oob_score=True  
)  
clf_bag.fit(X_train, y_train)
```

```
print(clf_bag.oob_score_)
```

```
0.9328125
```

```
pred = clf_bag.predict(X_test)  
print(accuracy_score(y_test, pred))
```

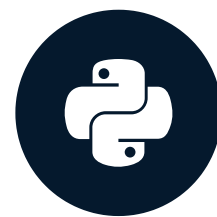
```
0.9625
```

# Now it's your turn!

ENSEMBLE METHODS IN PYTHON

# Bagging parameters: tips and tricks

ENSEMBLE METHODS IN PYTHON



**Román de las Heras**

Senior Data Scientist, Chartboost

# Basic parameters for bagging

## BASIC PARAMETERS

- `base_estimator`
- `n_estimators`
- `oob_score`
  - `est_bag.oob_score_`

# Additional parameters for bagging

## ADDITIONAL PARAMETERS

- `max_samples` : the number of samples to draw for each estimator.
- `max_features` : the number of features to draw for each estimator.
  - Classification  $\sim \sqrt{\text{number\_of\_features}}$
  - Regression  $\sim \text{number\_of\_features} / 3$
- `bootstrap` : whether samples are drawn with replacement.
  - True --> `max_samples = 1.0`
  - False --> `max_samples < 1.0`

# Random forest

## Classification

```
from sklearn.ensemble import RandomForestClassifier

clf_rf = RandomForestClassifier(
    # parameters...
)
```

## Regression

```
from sklearn.ensemble import RandomForestRegressor

reg_rf = RandomForestRegressor(
    # parameters...
)
```

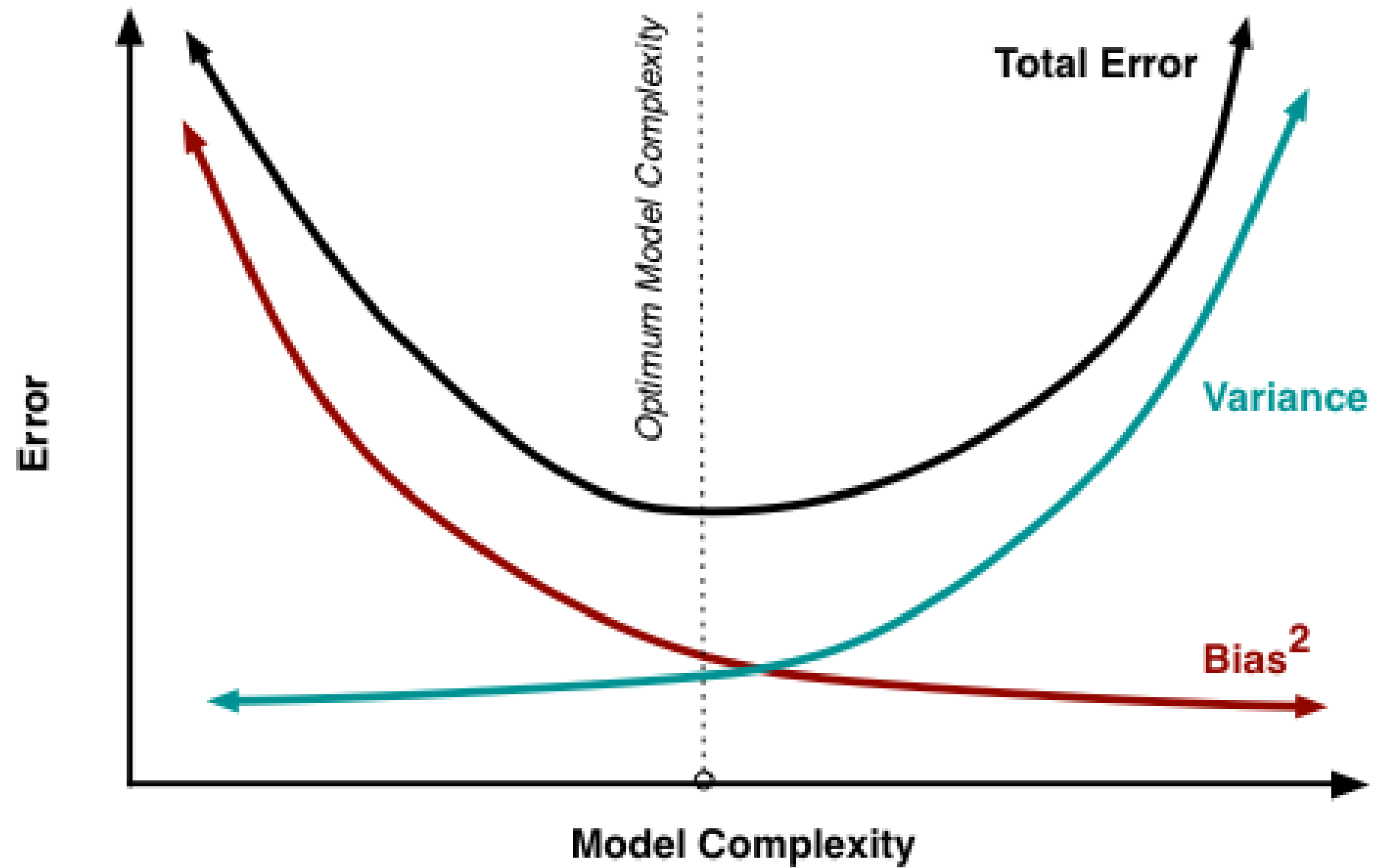
## Bagging parameters:

- `n_estimators`
- `max_features`
- `oob_score`

## Tree-specific parameters:

- `max_depth`
- `min_samples_split`
- `min_samples_leaf`
- `class_weight` ( "balanced" )

# Bias-variance tradeoff



**Let's practice!**  
ENSEMBLE METHODS IN PYTHON