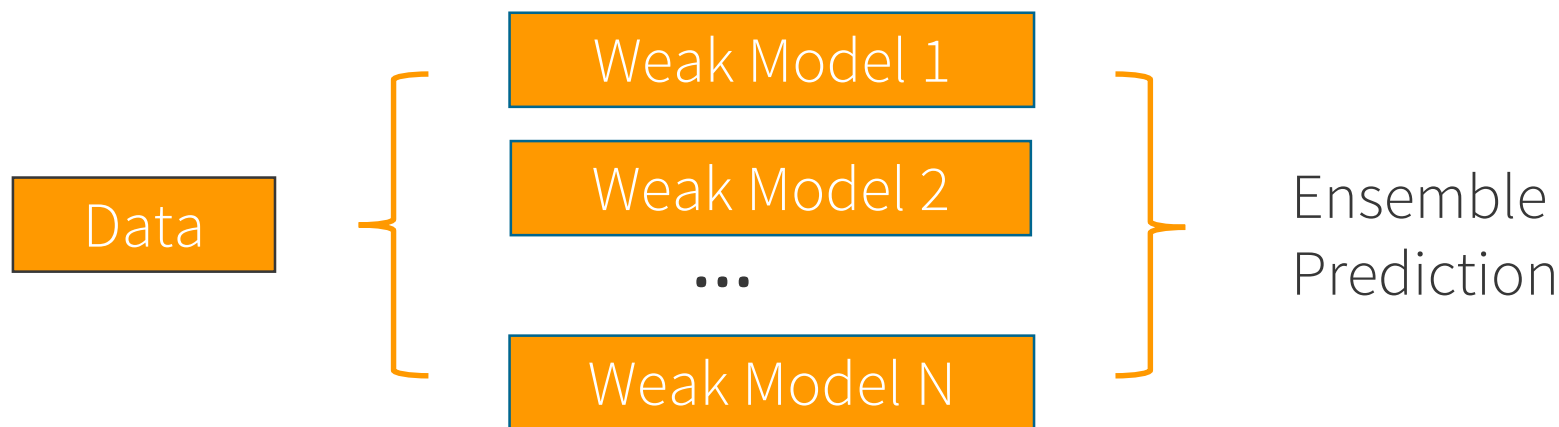


Ensemble Methods: Bagging

Ансамблевое обучение (Ensemble Learning)

Ансамблевые методы создают сильную модель, комбинируя прогнозы **нескольких слабых моделей** (также называемых **слабыми обучающимися** или **базовыми оценками**), построенных с заданным набором данных и заданным алгоритмом обучения.



Будем рассматривать модели ансамбля Bagging и Boosting.

Bagging (Bootstrap Aggregating)

Bagging (Bootstrap Aggregating) method:

- Произвольно возьмите N образцов фиксированного размера из обучающего набора (с заменой) - метод **bootstrap**

Example: given data [1, 2, 3, 4, 5, 6, 7, 8, 9], образцы размера 6:

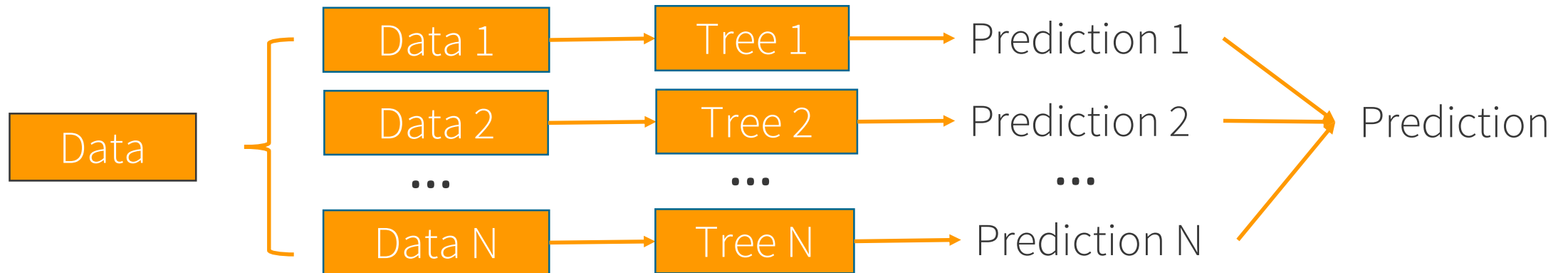
[1, 1, 2, 4, 9, 9]; [2, 4, 5, 5, 7, 7]; [1, 1, 1, 1, 1, 1]; [1, 2, 4, 5, 7, 9]

- Создавайте **независимые** оценщики (модели) одного типа для каждого подмножества
- Результат работы ансамбля: голосование по большинству (или взвешенное) или усреднение прогнозов всех оценщиков (для задачи регрессии).

Bagging trees: Random Forest

Random Forest: Bagging Decision Trees

- Создайте случайные подмножества (с заменой) из исходного набора данных
- Постройте **дерево решений** для каждого начального подмножества
- Объедините прогнозы из каждого дерева для окончательного прогноза



Random Forest in sklearn

RandomForestClassifier: sklearn Random Forest classifier (есть версия для Regressor) -

```
.fit(), .predict()
```

```
RandomForestClassifier(n_estimators=100,  
max_samples=None, max_features='auto',  
criterion='gini', max_depth=None, min_samples_split=2,  
min_samples_leaf=1, class_weight=None)
```

Полный список параметров гораздо больше.

Bagging in sklearn

BaggingClassifier: sklearn общий интерфейс для bagging который может быть предоставлен любым *base_estimator* - `.fit(), .predict()`

BaggingClassifier(*base_estimator=None, n_estimators=10, max_samples=1.0, bootstrap=True*)

Полный список параметров гораздо больше.

Hyperparameter Tuning

Настройка гиперпараметров (Hyperparameter Tuning)

- **Hyperparameters** это параметры алгоритмов машинного обучения, которые влияют на структуру алгоритмов и производительность (качество) моделей.

Примеры hyperparameters:

- **K Nearest Neighbors:** n_neighbors, metric
 - **Decision trees:** max_depth, min_samples_leaf, class_weight, criterion
 - **Random Forest:** n_estimators, max_samples
 - **Ensemble Bagging:** base_estimator, n_estimators
- **Hyperparameter tuning** (настройка гиперпараметров) ищет лучшую комбинацию гиперпараметров (комбинацию, которая максимизирует качество модели).

Grid Search in sklearn

GridSearchCV: **sklearn** базовый метод hyperparameter tuning, находит оптимальное сочетание гиперпараметров путем перебора в течение определенных значений параметров -

`.fit(), .predict()`

GridSearchCV(*estimator, param_grid, scoring=None*)

Example: Hyperparameters for a Decision Tree:

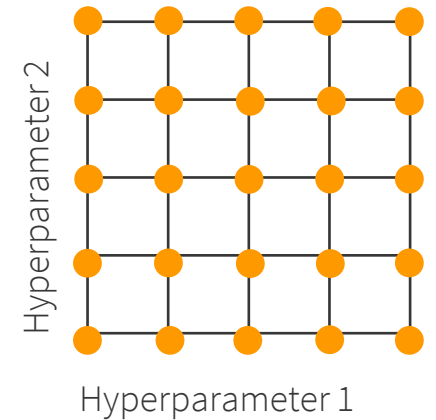
```
param_grid = {max_depth: [5, 10, 50, 100, 250],  
              min_samples_leaf: [15, 20, 25, 30, 35]}
```



Total hyperparameters combinations

$$5 \times 5 = 25$$

[5, 15], [5, 20], [5, 25], [10, 15], ...



Randomized Search in sklearn

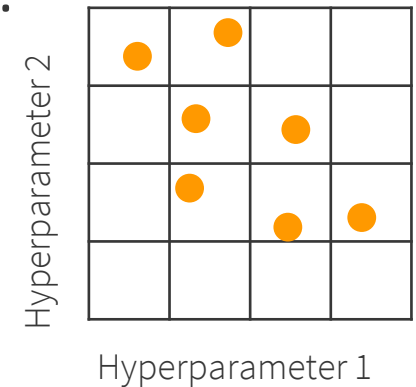
RandomizedSearchCV: рандомизированный поиск по гиперпараметрам

- Выбирает фиксированное количество (заданное параметром `n_iter`) случайных комбинаций значений гиперпараметров и пробует только их.
- Возможна выборка из распределений (используется выборка с заменой), если в качестве распределения задан хотя бы один параметр.

RandomizedSearchCV(*estimator, param_distributions,*
n_iter=10, scoring=None)

Example: Hyperparameters for a Decision Tree:

```
param_grid = {max_depth: [5, 10, 50, 100, 250],  
              min_samples_leaf: uniform(15,35,5)}
```



Байесовский поиск (Bayesian Search)

- Метод байесовского поиска (**Bayesian Search**) отслеживает предыдущие оценки гиперпараметров и строит вероятностную модель.
- Он пытается сбалансировать **исследование** (неопределенный набор гиперпараметров) и **реализацию** (гиперпараметры с хорошей вероятностью должны быть оптимальными).
- Предпочитает точки рядом с теми, которые хорошо сработали.

Data Preprocessing with Pipeline (sklearn)

Transformers in sklearn

- SimpleImputer, StandardScaler, MinMaxScaler, LabelEncoder, OrdinalEncoder, OneHotEncoder, и CountVectorizer принадлежат к классу transformers sklearn, у всех есть:
 - `.fit()` method: изучает преобразование из **обучающего** набора данных
 - `.transform()` method: применяет преобразование к любому набору данных (**training, validation, test**) для предварительной обработки

К обучающему набору также можно применить `.fit_transform()`

ColumnTransformer in sklearn

ColumnTransformer: применяет преобразователи к столбцам массива или pandas DataFrame – `.fit(), .transform()`

- Позволяет **отдельно преобразовывать** разные столбцы или подмножества входных столбцов (числовые, категориальные, текстовые).
- Функции, генерируемые каждым преобразователем, будут **объединены**, чтобы сформировать **единое пространство функций**.
- Это полезно для смешанных наборов табличных данных, чтобы объединить несколько механизмов извлечения признаков или преобразований в один преобразователь.

ColumnTransformer and Pipeline

```
numerical_processing = Pipeline([
    ('num_imputer', SimpleImputer(strategy='mean')),
    ('num_scaler', MinMaxScaler())])

categorical_processing = Pipeline([
    ('cat_imputer', Imputer(strategy='constant', fill_value='missing')),
    ('cat_encoder', OneHotEncoder(handle_unknown='ignore'))])

processor = ColumnTransformer(transformers=[
    ('num_processing', numerical_processing, ('feature1', 'feature3')),
    ('cat_processing', categorical_processing, ('feature0', 'feature2'))])

pipeline = Pipeline([('data_processing', processor),
    ('estimator', KNeighborsClassifier())])

pipeline.fit(X_train, y_train)
predictions = pipeline.predict(X_test)
```

