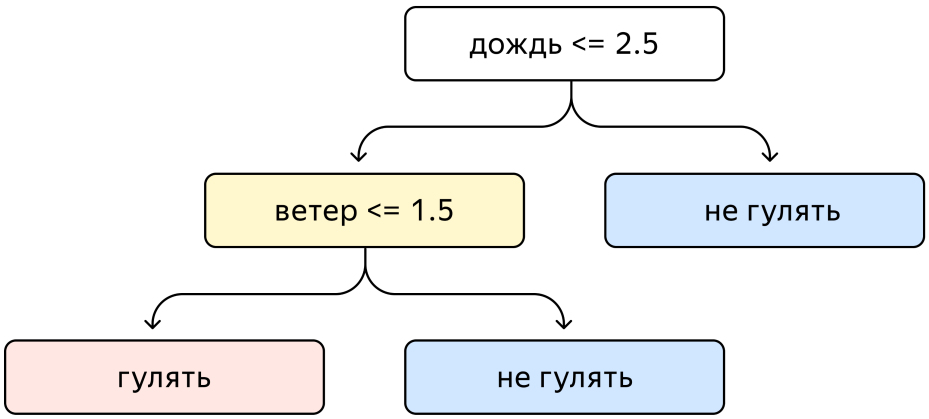
Дерево решений и гиперпараметры

Дерево решений (англ. decision tree) — модель МО иерархической структуры из правил вида «Если... то...». В машинном обучении подобные правила создаются автоматически в процессе обучения.

Элементы дерева решений:
- корневой узел — вершина дерева;
- узлы (узлы принятия решений) — элементы с правилами;
- конечные узлы (листья) — узлы, в которых нет правил;
- глубина дерева — количество уровней с узлами, в которых есть правила.



Гиперпараметры дерева решений:

- max_depth — максимальная глубина дерева. Гиперпараметр фиксирует количество уровней с правилами. В модели с базовыми настройками этот параметр равен None. Это значит, что глубина дерева может быть какой угодно, из-за чего модель переобучается. Если зададим этот гиперпараметр целым числом, ниже этого уровня дерево строиться не будет.

- min_samples_split — минимальное число объектов в узле, при котором он разделится на новые узлы с объектами разных классов. Если объектов меньше указанного числа, то разделения на новые узлы не будет. Значение гиперпараметра по умолчанию — 2. Если в узле два объекта — они могут разделиться ещё на два узла.

- min_samples_leaf — минимальное число объектов в листе. Гиперпараметр отвечает за количество объектов в одном листе. Значение по умолчанию — 1, так в листе сможет находиться всего один объект. Если сделать значение больше, листы не будут формироваться, если в них попадёт недостаточное число объектов.

Пример использования дерева решений

```
from sklearn.tree import DecisionTreeClassifier

# объявляем классификатор
model = DecisionTreeClassifier(
    max_depth=5, # максимальная глубина
    min_samples_split=2, # минимум объектов, при котором узел делится на подузлы
    min_samples_leaf=1, # минимальное число объектов в листе
    random_state=42 # фиксирование случайности
)

model.fit(X_train, y_train) # обучаем модель

y_pred = model.predict(X_test) # получаем предсказания
```

Оптимизация гиперпараметров

Параметры — это настройки модели, которые подбираются в процессе обучения модели на данных. Их подбирает сама модель, когда вы вызываете метод fit().

Гиперпараметры — настройки модели, которые фиксируются вручную до начала обучения и в процессе обучения никак не меняются.

Способы оптимизации гиперпараметров

Способ	Суть способа	Диапазон значений гиперпараметров	Скорость работы	Ускорение вычислений и воспроизводимость
`GridSearchCV`	Осуществляет перебор по сетке, все указанные значения гиперпараметров будут перебираться	Лучше перебирать небольшие диапазоны значений	Занимает больше времени по сравнению с другими способами	Можно ускорить вычисления с помощью `n_jobs=-1`
`RandomizedSearchCV`	Перебирает гиперпараметры случайно, `n_iter` указывает, сколько комбинаций перебрать	Лучше перебирать большие диапазоны значений	Занимает меньше времени по сравнению с другими способами	Можно ускорить вычисления с помощью `n_jobs=-1`
`OptunaSearchCV`	Осуществляет перебор гиперпараметров с помощью байесовской оптимизации, можно ограничить количество запусков с помощью `n_trials`	Лучше перебирать средние диапазоны значений	Занимает меньше времени, чем `GridSearchCV`, но больше времени, чем `RandomizedSearchCV`	Если ускорять вычисления с помощью `n_jobs=-1`, потеряется стабильность результатов

Дерево решений и гиперпараметры

- **GridSearchCV**

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=42)

# словарь с гиперпараметрами
parameters = {
    'min_samples_leaf': range(1, 3),
    'max_depth': range(1, 3)
}

gs = GridSearchCV(
    model, # модель для подбора гиперпараметров
    parameters, # словарь со значениями гиперпараметров
    n_jobs=-1, # количество вычислительных мощностей
    cv=5, # тип кросс-валидации
    scoring='roc_auc' # метрика для оценки качества обученных моделей
)

gs.fit(X_train, y_train)
```

Дерево решений и гиперпараметры

- **RandomizedSearchCV**

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=42)

# словарь с гиперпараметрами
parameters = {
    'min_samples_leaf': range(1, 3),
    'max_depth': range(1, 3)
}

rs = RandomizedSearchCV(
    model, # модель для подбора гиперпараметров
    parameters, # словарь со значениями гиперпараметров
    n_jobs=-1, # количество вычислительных мощностей
    cv=5, # тип кросс-валидации
    scoring='roc_auc', # метрика для оценки качества обученных моделей
    n_iter=10, # количество комбинаций гиперпараметров
    random_state=42 # фиксирование случайности
)

rs.fit(X_train, y_train)
```

Дерево решений и гиперпараметры

- **OptunaSearchCV**

```
from optuna import distributions
from optuna.integration import OptunaSearchCV
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=42)

# словарь с гиперпараметрами
parameters = {
    'min_samples_leaf': distributions.IntDistribution(1, 4),
    'max_depth': distributions.IntDistribution(1, 4),
}

oscv = OptunaSearchCV(
    model, # модель для подбора гиперпараметров
    parameters, # словарь со значениями гиперпараметров
    cv=5, # тип кросс-валидации
    scoring='roc_auc', # метрика для оценки качества обученных моделей
    n_trials=10, # количество итераций поиска
    random_state=42 # фиксирование случайности
)

oscv.fit(X_train, y_train)
```