Tree and Ensemble Methods

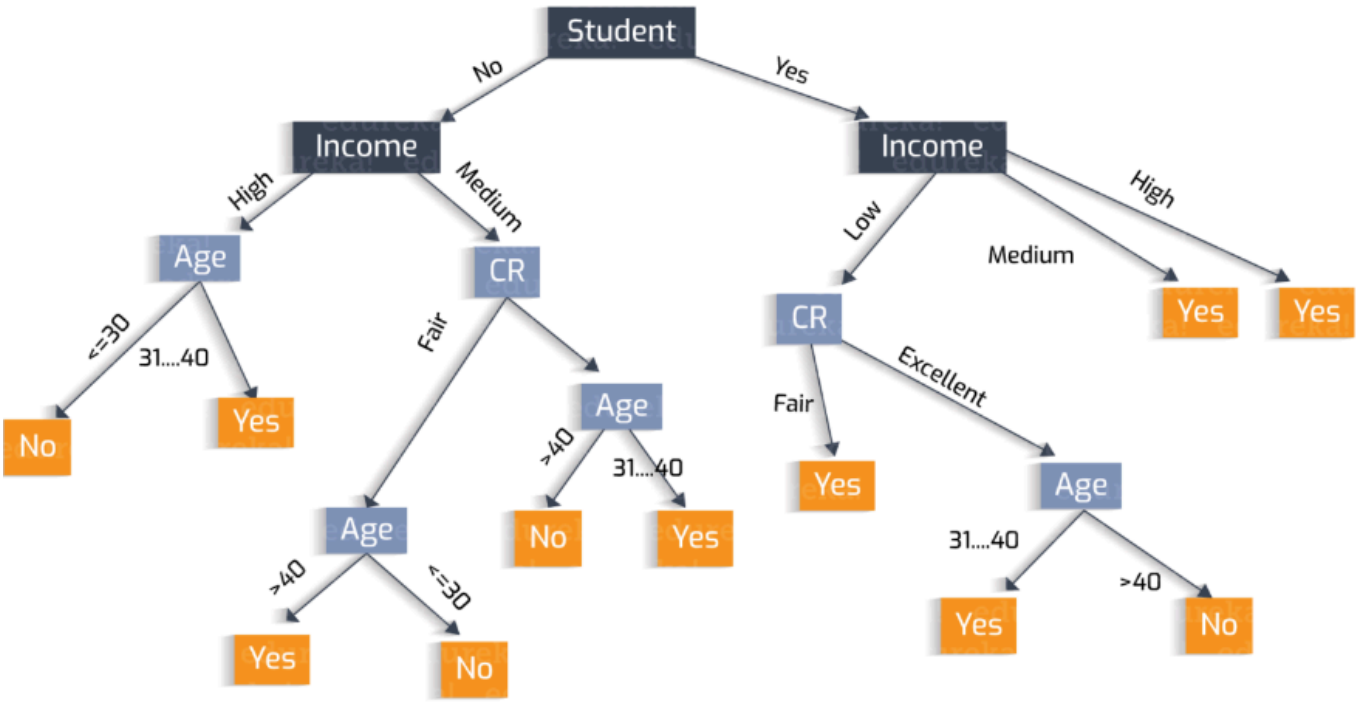
екстерполация да проверя какво е

Ентропия - ключово понятие, използвано в машинното обучение, за да измери степента на несигурност или хаос в данните. При Decision trees тя помага да се определи колко добре дадено разделяне на данните (чрез тест на характеристика) води до по-добро разделяне на класовете.

1. Decision Trees

1.1 Определение

Модел, който представлява поредица от решения или правила, организирани в йерархична структура, подобна на дърво. Всяко вътрешно възелче (възел) представлява тест върху определена характеристика (функция) на данните, а всеки клон (глава) от възела води до нови възли или до крайни резултати (листа).



1.2 Основни компоненти:

1. **Корен(Root Node):** Началната точка на дървото, където започва процесът на вземане на решения.

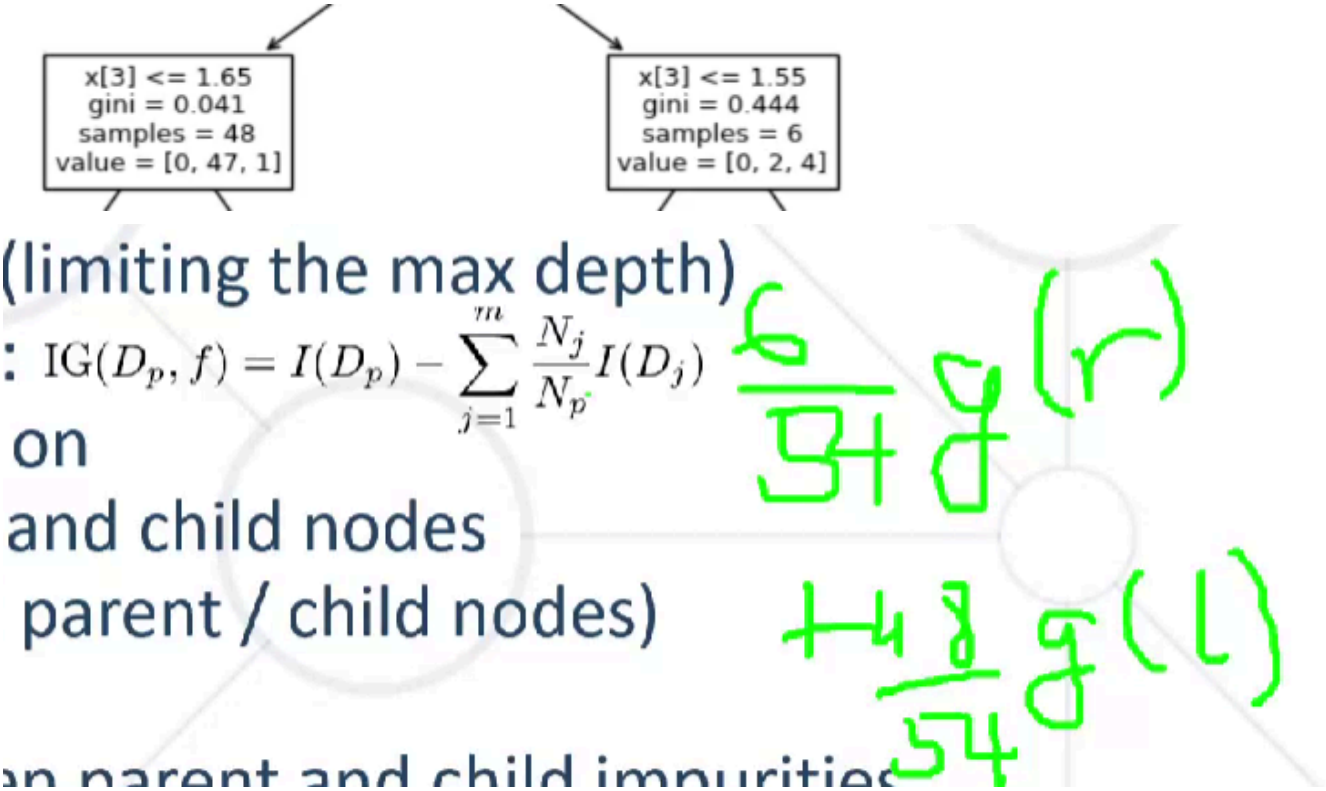
2. **Вътрешни възли (Internal Nodes):** Представляват тестове или условия върху характеристиките на данните.
3. **Клонове (Branches):** Свързват възлите и показват резултата от теста.
4. **Листа (Leaf Nodes):** Крайните възли, които дават окончателното решение или прогнозата.

1.3 Сумарната ентропия

Сумарната ентропия показва колко добре дадено разделение намалява неопределеността в класификациите. Целта на дървото на решенията е да минимизира тази стойност при всяко разделяне, за да се стигне до възможно най-добро разделяне, където ентропията е нулева (чисти класове).

1.4 IG - Information gain

IG (Information Gain), или **информационно печалба**, е мярка, използвана в дърветата на решенията, за да се определи коя характеристика (променлива) трябва да бъде използвана за разделяне на данните в следващия възел. Информационната печалба показва колко добре дадена характеристика разделя данните, като измерва намаляването на **ентропията** след разделянето. Колкото по-висока е информационната печалба, толкова по-добре характеристиката разделя данните и толкова по-полезна е тя за вземането на решения.



2. Decision Forest

2.1 Определение

Мощен метод в областта на машинното обучение, използван както за класификация, така и за регресия. Той съчетава множество decision trees с цел подобряване на точността и предотвратяване на overfitting.

2.2 Как работи Decision Forest?

1. Създаване на множество дървета: Моделът създава много Decision trees, като всяко дърво се обучава на различна извадка от данните (използвайки bootsrap метод – случайно избиране с повторения).
2. **Разнообразие чрез случайност**: При всяко дърво, при избора на най-доброто разделение на данните, се използва случайно подмножество от характеристиките (функциите). Това води до разнообразие между дърветата, което подобрява общата производителност на модела.
3. **Гласуване (voting) или усредняване**: За класификационни задачи, всяко дърво дава свой „глас“ за класа, а моделът избира класа с най-много гласове. За регресионни задачи, резултатите от дърветата се усредняват.

2.3 Предимства на Decision Forest:

- **Висока точност**: Комбинирането на множество дървета води до по-точни предвиждания.
- **Устойчивост на пренастройване**: Моделът е по-малко склонен към пренастройване в сравнение с отделните дървета.
- **Гъвкавост**: Може да се използва както за класификация, така и за регресия.
- **Обработка на големи данни**: Ефективен при работа с големи и сложни набори от данни.

2.4 Permutation feature importance

Пермутационната важност на характеристиките е техника за оценка на значимостта на всяка характеристика (feature) в моделите за машинно обучение. Този метод измерва колко се влошава производителността на модела, когато стойностите на дадена характеристика са случайно разбъркани (пермутирани). Ако разбъркването на

характеристиката води до значително влошаване на производителността, това означава, че характеристиката е важна за модела.

3. Ensemble methods

3.1 Определение

Начин да комбинираме слаби алгоритми (такива с high bias).

Ensemble methods - в машинното обучение (ML) представляват техника, при която няколко модели се комбинират, за да постигнат по-добри резултати, отколкото всеки от тях поотделно. Тази стратегия се основава на идеята, че комбинирането на множество слаби модели може да доведе до силен модел, който е по-точен, устойчив на грешки и по-добре генерализира върху нови данни.

3.2 Основни концепции в Ensemble methods:

1. **Слаби и силни модели**:
   - **Слаб модел**: Това е модел, който има малко по-добра производителност от случайно познаване, като обикновено има сравнително ниска точност.
   - **Силен модел**: Той се създава чрез комбиниране на множество слаби модели и се отличава с висока точност и по-добра обобщаваща способност.
2. **Разнообразие на моделите**: Ensemble methods разчитат на разнообразието на моделите. Всеки от моделите в ансамбъла трябва да има различно поведение, за да може комбинираният резултат да бъде по-точен. Модели, които допускат различни грешки, се допълват взаимно.
3. **Комбиниране на резултати**: Важно е как се комбинират предсказанията на различните модели. Основните методи за това включват:
   - **Гласуване (Voting)**: Мнението на всеки модел се взима под внимание и се избира този клас, който е предложен от мнозинството.
   - **Средно аритметично (Averaging)**: Ако става въпрос за регресия, предсказанията на моделите се усредняват.

3.3 Видове ensemble methods

1. **Bagging (Bootstrap Aggregating)**
   - **Основна идея**: Методът Bagging се основава на създаване на множество подмножества от оригиналния набор от данни, чрез метода на бутстрап

(случайно избиране на данни с повторение). За всяко подмножество се обучава отделен модел, и след това предсказанията на тези модели се обединяват.
   - **Основна цел**: Намаляване на вариацията на модела и предотвратяване на пренастройването (overfitting).
   - **Пример**: Random Forest е класически пример за Bagging, където се използват множество решаващи дървета (Decision Trees), обучени върху различни подмножества от данните.
2. **Voting Classifier (гласуване)**
   - **Основна идея**: В този метод няколко различни модела гласуват за всеки пример от данните. Мнението на мнозинството решава финалната класификация. Може да се използва хард гласуване (където се взима решението на най-многочисления клас) или софт гласуване (където се взимат предвид вероятностите).
   - **Основна цел**: Преодоляване на грешките на отделни модели чрез комбиниране на техните решения.
   - **Пример**: Използването на SVM, логистична регресия и решаващи дървета заедно в един ансамбъл.
3. **Stacking (Stacked Generalization)**
   - **Основна идея**: Stacking използва мета-модел, който комбинира предсказанията на няколко различни базови модела (като решаващи дървета, логистична регресия и др.). Мета-моделът се обучава на изходните резултати на тези базови модели, за да направи финалното предсказание.
   - **Основна цел**: Използване на предимствата на различни модели, като всеки базов модел може да се справя по-добре с различни аспекти на данните.
   - **Пример**: Може да се използват няколко различни ML модела, а финалното предсказание да се извършва чрез алгоритъм като линейна регресия.
4. **AdaBoost (Adaptive Boosting)**
   Автоматично се адаптира към грешките на слаби класификатори, като им придава тежест в зависимост от тяхната точност.
5. **GradientBoosting**
   - Както и в AdaBoost, в Gradient Boosting се създават множество последователни слаби модели, като всеки нов модел е насочен към коригирането на грешките на предишния.
   - Всеки следващ модел „усилва“ предсказвателната сила на предходните, така че крайната комбинация от модели да има висока точност.
   - За разлика от AdaBoost, където фокусът е върху примери, които са грешно класифицирани, Gradient Boosting се фокусира върху минимизирането на **функцията на загуба (loss function)**, като използва метод, наречен **градиентен спуск**.

- Алгоритъмът се стреми да минимизира тази функция на загуба чрез поредица от стъпки, като всеки нов модел прави предсказания върху остатъчните грешки (residuals), направени от предишния модел.