10. **Учене на група от модели**

• **Комбиниране на множество модели /Ensemble learning/-** Използваме **няколко модела**, вместо един, за да **подобрим точността** и **устойчивостта** на предсказанията. *Пример:* Вместо да разчитаме само на едно дърво за решения, използваме 100 и вземаме мнозинството.

Bagging(Bootstrap Aggregating) -Метод, при който обучаваме много модели върху случайни подмножества от данните и ги комбинираме (например чрез гласуване). *Пример:* Обучаваме 50 дървета на различни случайни части от набора и вземаме средното на резултатите.

Разлагане на пристрастие и дисперсия

- **Пристрастие (bias):** Грешка от прекалено опростен модел.
- **Дисперсия (variance):** Грешка от прекалено сложен модел, чувствителен към шума.

***Пример:* Bagging намалява дисперсията, като изглажда резултатите от различни модели.**

Bagging с разходи- Вариант на bagging, който взема предвид цената на грешките. По-важните грешки (напр. в медицината) получават по-голяма тежест. *Пример:* По-сериозно се наказва сгрешена диагноза за болен пациент, отколкото фалшива тревога.

Рандомизация -Добавяне на случайност в обучението, за да се повиши разнообразието между моделите. *Пример:* Вместо винаги да използваме всички характеристики, позволяваме на всяко дърво да избира само част от тях.

Random Forests (Случайни гори) -Много дървета, обучени чрез bagging + случайни характеристики, които гласуват за финален резултат. *Пример:* Класифицираме дали имейл е спам чрез 100 случайни дървета.

Rotation Forests-Метод, който завърта (трансформира) характеристиките чрез PCA преди да обучи всяко дърво — повишава разнообразието и точността. *Пример:* Дърветата виждат различни "ъгли" на данните, което дава по-добра представа.

Boosting- Обучаваме модели един след друг, като всеки следващ оправя грешките на предишния. *Пример:* Първият модел се справя зле с някои данни – вторият се фокусира точно върху тях.

AdaBoost (Adaptive Boosting) - Boosting, който дава по-голяма тежест на грешно класифицираните примери, за да се поправят по-нататък. *Пример:* Имейл погрешно маркиран като "не спам" получава по-голяма тежест при следващия модел.

Силата на boosting- Boosting често постига по-висока точност от всеки индивидуален модел, но може да се преобучи, ако не се контролира добре.

Адитивна регресия- Boosting, при който всеки модел прибавя (адитивно) корекция към предсказанията на предишните модели.

Пример: Прогнозираме цена на жилище, като всеки нов модел добавя малка корекция към предишната стойност.

Числова прогноза (в контекста на boosting)- Boosting се използва не само за класификация, а и за регресия – предсказване на числови стойности. *Пример:* Прогноза на температура по дни с boosting регресия.

Адитивна логистична регресия-Boosting подход за класификация, който използва логистична функция и адитивни модели. *Пример:* Класификация дали клиент ще купи продукт, с използване на логистична функция + boosting.

Интерпретируеми ансамбли- Комбинирани модели, които остават разбираеми за хората (не като "черни кутии").

Option Trees- Дървета, които на възел може да имат няколко алтернативни поддървета, не само един път. *Пример:* Ако условието "възраст > 30" не е ясно, се разглеждат и други разклонения едновременно.

Алтернативни дървета за решения (Alternating Decision Trees)- Комбинират решаващи възли и оценъчни възли, като се преминава през всички приложими пътища, а не само един. *Пример:* Имаме множество допринасящи фактори за всяко решение, не само една последователност от избори.

Логистични моделни дървета-Комбинация между дървета и логистична регресия – във всеки лист се прилага логистичен модел вместо просто клас. *Пример:* В края на дървото се прави по-прецизна прогноза чрез логистична регресия.

Stacking (Натрупване)-Обучаваме различни модели и след това обучаваме друг модел (мета-модел), който комбинира изхода им. *Пример:* Имаме SVM, Decision Tree и KNN, а логистичен модел решава как да ги комбинира най-добре.

Комбиниране на множество модели

• Основна идея: изграждане на различни „експерти“, които гласуват.
• Предимство: Често подобрява предсказателната производителност.
• Недостатък: Обикновено произвежда изход, който е много труден

за анализ.• Но: има подходи, които целят да създадат една разбираема структура.

Bagging-Комбиниране на прогнози чрез гласуване/усредняване. Всеки модел получава равно тегло. „Идеализирана“ версия: Вземане на няколко тренировъчни набора с размер n (вместо само един тренировъчен набор с размер n). Изграждане на класификатор за всеки тренировъчен набор. Комбиниране на прогнозите на класификаторите. Схемата за обучение е нестабилна, почти винаги подобрява производителността. Нестабилен обучаем: малка промяна в тренировъчните данни може да доведе до голяма промяна в модела (напр. при обучение на дървета за решения).

Bagging класификатори-Нека n е броят на екземплярите в тренировъчните данни. За всяка от t итерации:
• Вземане на n екземпляра от тренировъчния набор (с връщане).
• Прилагане на алгоритъма за обучение върху извадката.
• Запазване на получения модел.
За всеки от t модела:
• Предсказване на класа на екземпляра с помощта на модела.
• Връщане на класа, който е предсказан най-често.

Рандомизация и random forests-Може да се рандомизира алгоритъмът за обучение вместо входа. Някои алгоритми вече имат случаен компонент: напр. начални тегла в невронна мрежа. Повечето алгоритми могат да бъдат рандомизирани, напр. алчни алгоритми:
• Избиране на N опции на случаен принцип от пълния набор от опции, след което избор на най-добрата от тези N опции.
• Напр.: избор на атрибути в дървета за решения.
• По-общо приложим от **bagging**: напр. можем да използваме случайни подмножества в класификатор за най-близки съседи.

• **Bagging** не работи със стабилни класификатори като тези за най-близки съседи.
• Може да се комбинира с **bagging**.
• Когато се използва с дървета за решения, това води до известния метод **random forest** за изграждане на ансамблови класификатори.

Boosting
• **Bagging** лесно може да се паралелизира, защото членовете на ансамбъла се създават независимо.
• **Boosting** е алтернативен подход.Също използва гласуване/усредняване. Но: тегли моделите според производителността. Итеративен: новите модели се влияят от производителността на предишно изградените. Насърчава новия модел да стане „експерт“ за екземпляри, погрешно класифицирани от по-ранни модели. Интуитивно оправдание: моделите трябва да са експерти, които се допълват взаимно. Съществуват много варианти на **boosting**, ние разглеждаме няколко.

Boosting с AdaBoost.M1-Присвояване на равно тегло на всеки тренировъчен екземпляр.За t итерации:
• Прилагане на алгоритъма за обучение върху претеглениия набор от данни, запазване на получения модел.
• Изчисляване на грешката e на модела върху претеглениия набор от данни. Ако $e=0$ или $e \geq 0.5$: Прекратяване на генерирането на модели.
• За всеки екземпляр в набора от данни: Ако е класифициран правилно от модела: Умножаване на теглото на екземпляра по $e/(1-e)$

• Нормализиране на теглото на всички екземпляри.
Класификация: Присвояване на тегло = 0 на всички класове.
• За всеки от t (или по-малко) модела: За класа, предсказан от този

модел, добавяне на −loge/(1−e) към теглото на този клас. Връщане на класа с най-високо тегло.

Коментари за AdaBoost.M1

• **Boosting** изисква тегла … но Може да се адаптира алгоритъмът за обучение … или Може да се приложи **boosting** без тегла: Пресъздаване на данни с вероятност, определена от теглата.
• Недостатък: не всички екземпляри се използват.
• Предимство: ако грешката e > 0.5, може да се пресъздаде отново.
• Алгоритъмът за **boosting AdaBoost.M1** произлиза от работа в теорията на изчислителното обучение.
• Теоретичен резултат: Грешката при обучение намалява експоненциално с извършването на итерации. Други теоретични резултати: Работи добре, ако базовите класификатори не са твърде сложни и Грешката им не става твърде голяма твърде бързо с увеличаване на итерациите.

Stacking - Въпрос: как да се изгради хетерогенен ансамбъл, състоящ се от различни типове модели (напр. дърво за решения и невронна мрежа)? Проблем: моделите могат да се различават значително по точност. Идея: за комбиниране на прогнозите на базовите обучаеми, вместо просто да се гласува, да се използва мета-обучаем (**meta learner**). В **stacking**, базовите обучаеми се наричат още модели от ниво-0 (**level-0 models**). Мета-обучаемият се нарича модел от ниво-1 (**level-1 model**). Прогнозите на базовите обучаеми са вход за мета-обучаемия. Базовите обучаеми обикновено са различни схеми за обучение. Предупреждение: не може да се използват прогнози върху тренировъчните данни за генериране на данни за модела от ниво-1! Вместо това се използва схема, базирана на кръстосана валидация (**cross-validation**).