

**Технически Университет – София**

**Факултет Приложна Математика и Информатика**

**Катедра Информатика**

**КУРСОВА РАБОТА ПО ПРИЛОЖЕН ИЗКУСТВЕН ИНТЕЛЕКТ**

**Тема:** Decision trees

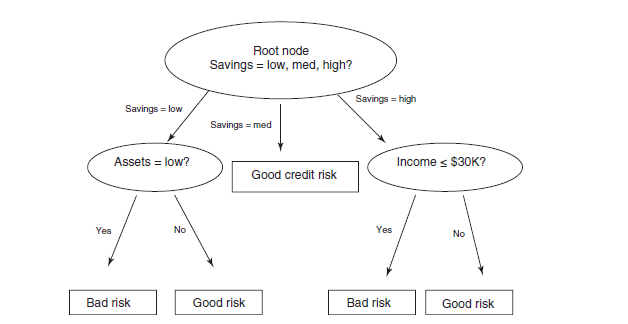
**Имена на студентите:** Тодор Пъков, Георги Донков

**Факултетни номера, групи:** 471218003, 471218026, 78 група

**Съдържание:**

1. Дърво на решенията
   1. Какво представлява дървото на решенията?
   2. Изисквания за използване на дърво на решенията
2. Метрики за намиране на най-добро разделяне
   1. Gini impurity
   2. Информационна печалба (Information Gain)
   3. Намаляване на дисперсията (Variance reduction)
   4. Мярка за доброта (Measure of “goodness”)
3. Алгоритъм CART (Classification and Regression Tree)
4. Алгоритъм C4.5
5. Дърво на решенията
   1. Какво представлява дървото на решенията?

Дървото на решенията е метод за класификация, който представлява конструкция от възли (nodes), които са свързани с клонове (branches). Конструкцията тръгва отгоре надолу, като най – отгоре имаме възел, наречен корен (root node), най – отдолу имаме възли, наречени листа (leaf nodes), а между корена и листата съществуват така наречените decision nodes.



На илюстрацията отгоре е показано дърво на решенията, което оценява кредитния риск, като класифицира даден кандидат с „добър“ или „лош“ кредитен риск в зависимост от няколко прогнозни променливи (predictor variables), а именно спестявания, които биват ниски, средни или високи; активи, които биват ниски или високи и доходи, които се класифицират като по – малко от тридесет хиляди долара или по – големи от тридесет хиляди долара.

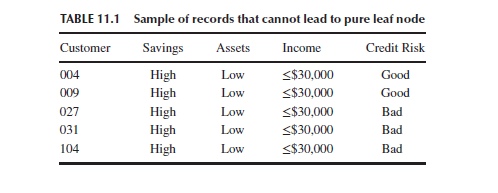
На дадената илюстрация коренът на дървото е всъщност decision node, който проверява дали за даден запис от набора данни (в случая кандидат за кредит) спестяванията са ниски, средни или високи. Според стойността на дадената прогнозна променлива се тръгва по някой от клоните на дървото.

Ако спестяванията са ниски, то ще тръгнем по най – левия клон на дървото, който води до друг decision node, в който се проверява дали активите са ниски. Ако са ниски – то кандидатът за кредит се класифицира с добър кредитен риск, а ако са високи – с лош кредитен риск (тук достигаме до листата на дървото и приключваме обхождането му).

Ако спестяванията са високи, то тръгваме по най – десния клон, който води до друг decision node, където се проверява дали приходите на кандидата са повече или по – малко от тридесет хиляди долара. Ако са повече, то кандидатът се класифицира с добър кредитен риск, а ако са по – малко – с лощ кредитен риск.

Последно остана да прегледаме случая, когато спестяванията са средни. Тогава ще тръгнем по средния клон на дървото, който ни свързва директно с едно от листата на дърветата, тоест кандидатите със средни спестявания директно се класифицират с добър кредитен риск, без да се проверяват други прогнозни променливи.

Все пак има случаи, в които чрез даден decision node не можем да достигнем до „чисто“ листо на дървото. Нека вземем за пример следния откъс от данни:



Ако тръгнем от корена на дървото, то ясно се вижда, че за всеки един от кандидатите ще тръгнем по най – десния клон, защото спестяванията на всички за високи. Най – десния клон ни свързва с decision node-а, който проверява дали доходите са по – малки от тридесет хиляди долара.

Имаме два случая, в които доходите са по – малки от тридесет хиляди, но кандидатите в набора от данни са класифицирани с добър кредитен риск. Останалите трима кандидати са класифицирани с лош кредитен риск.

Все пак в случая класифицираните с лош кредитен риск преобладават и следователно можем да достигнем до листо на дървото, което да класифицира кандидатите за кредит с доходи по – малки от тридесет хиляди долара с лош кредитен риск.

Проблемът е, че това развитие ще има 60% успех. В случая можем да се опитаме да разклоним още повече дървото, като първо проверим доходите на кандидатите, а след това добавим още decision nodes, които да проверяват спестяванията или акциите от най – дясната страна на дървото.

Наблюдавайки данните за кандидатите виждаме, че всички от тях имат еднаква стойност за спестяванията и акциите, т.е. няма смисъл от това да правим разклонения. В такъв случай листото на дървото, до което сме достигнали се нарича diverse leaf.

* 1. Изисквания за използване на дърво на решенията

Следните изисквания трябва да бъдат спазени преди използването на алгоритъм за изграждане на дърво на решенията:

1. Алгоритмите, които изграждат дървета на решенията, изискват предварително класифицирани променливи. Трябва да предоставим набор от данни, чрез който ще обучим алгоритъма, предоставяйки стойностите на класифицираните променливи.
2. Наборът от данни трябва да е богат и да предоставя на алгоритъма множество комбинации от различни стойности за класифицираните променливи.
3. Целевите променливи трябва да приемат стойности, които могат да се определят като принадлежащи или непринадлежащи на даден клас.

Важен е въпросът защо например в по – горния пример за корен на дървото е избрана проверката на атрибута спестявания. Отговорът се крие в това, че дървото на решенията се изгражда по начин, по който листата му ще са най „чисти“. Това се случва чрез измерване на чистотата на листата (leaf node purity) с различни алгоритми.

1. Метрики за намиране на най-добро разделяне

Алгоритмите за конструиране на дърво на решенията обикновено от горе надолу, като на всяка стъпка се избира променливата, която най-добре разделя множеството от данни. Различните алгоритми имат специфични подходи за намирането на това „най-добро“ разделяне. Обикновено се измерва хомогенността на целевата променлива в отделните подмножества. Изчисленията се прилагат към всяко кандидат-множество и резултатите се комбинират, като така осигуряват информация за качеството на съответното разделяне.

* 1. Gini impurity

Използва се от алгоритъма CART и измерва колко често случайно избран елемент от множеството ще бъде грешно определен (класифициран), ако е бил случайно класифициран спрямо разпределението на класовете в подмножеството. Може да се изчисли като сумират вероятностите даден запис от клас да бъде избран, умножени по вероятността за грешка при класифицирането. Функцията достига своя минимум – нула, когато всичките записи във възела принадлежат на един клас.

За изчисление на ф-ята за множество от елементи с J класа и – пропорцията от елементи, принадлежащи към клас , може да се използва формулата:

D:\Documents\ПИИ\{_displaystyle _operatorname {I} _{G}(p)=_sum _{i=1}^{J}_left(p_{i}_sum _{k_neq i}p_{k}_right)=_sum _{i=1}^{J}p_{i}(1-p_{i})=_sum _{i=1}^{J}(p_{i}-{p_{i}}^{2})=_sum _{i=1}^{J}p_{i}-_sum _{i=1}^{J}{p_{i}}^{2}=1-_sum _{i=1}^{J}{p_{.png

* 1. Информационна печалба (Information Gain)

Използва се при алгоритмите ID3, C4.5 и C5.0. Основава се на концепциите за ентропия и информационно съдържание от теорията на информацията. Разгледана е по-подробно в частта за C4.5.

* 1. Намаляване на дисперсията (Variance reduction)

Имплементирана е за първи път в CART. Обикновено се използва когато целевата променлива е непрекъсната (регресионно дърво), тъй като използването на други метрики би изисквало дискретизация на данните. Намаляването на дисперсията в даден връх N се дефинира като общото намаляване на дисперсията на целевата променлива, породено от извършване на разделяне при N.

* 1. Мярка за „доброта“ (Measure of “goodness”)

Използвана е в CART през 1984 г. Това е функция, имаща за цел да оптимизира баланса между способността на възможните разделяния да създават чисти деца (с елементи от един клас) с възможността им да създават деца с еднакъв размер (брой записи). Процесът се повтаря за всеки „нечист“ възел до създаване на цялото дърво. Самата ф-я , където s – едно възможно разделяне при възел t, изглежда така:

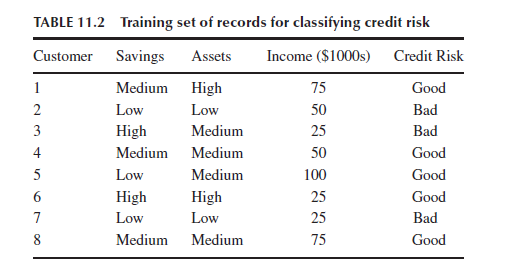
Във формулата и са съответно левия и десния детски възел на t, в резултат от разделяне s, и са пропорциите на елементите от t в и , а и са пропорциите на елементите от клас j в и .

Функцията се увеличава, когато и двата й компонента - и също нарастват. Вторият ще има голяма стойност, когато разстоянията между и са максимални за всеки клас или иначе казано, когато пропорциите на елементите в дъщерните възли за всеки възможен клас са колкото се може по-различни. Теоретичният максимум на тази част от е равен на самия брой класове, налични в данните. Другият компонент - достига максимум, когато пропорциите на елементите в левия и десния дъщерен възел са равни ().

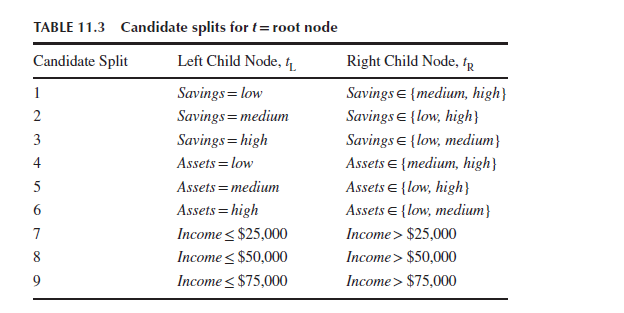
От казаното дотук може да се заключи, че дърво, използващо ф-ята , ще предпочита разделяния, при които дъщерните възли са хомогенни спрямо всички класове и имат приблизително еднакъв брой елементи.

1. Алгоритъм CART (Classification and Regression Tree)

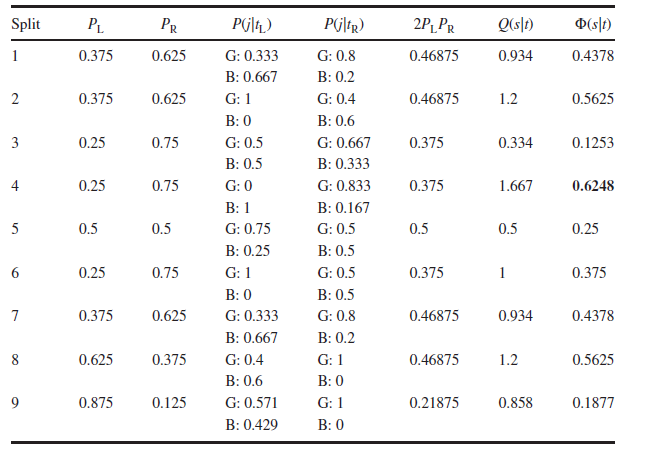
Методът на класификационните и регресионни дървета CART е предложен през 1984 г. от Л. Бреймън, Дж. Фрийдмън, Р. Олшен и С. Стоун. Съществуват различни варианти на CART, напр. CHAID, ExhaustiveCHAID и др. Като регресионна техника CART методът се определя като рекурсивно-разделяща регресия. Целта е разделяне на данните в относително хомогенни (с ниско стандартно отклонение или с минимална обща грешка по метода на най-малките квадрати) крайни възли и получаване на средна наблюдавана стойност при всеки краен възел като прогнозна стойност. За намиране на най – доброто разделяне в дървото на решенията, което се изгражда от CART алгоритъма, можем да използваме мярката за доброта (measure of goodness) или gini impurity, които са обяснени подробно по – нагоре в текущия документ.

Нека разгледаме един пример как чрез CART алгоритъма можем да изградим дърво на решенията при даден набор от данни използвайки метриката за намиране на най – добро разделяне мярка за доброта (measure of goodness). Нека наборът от данни е следният:

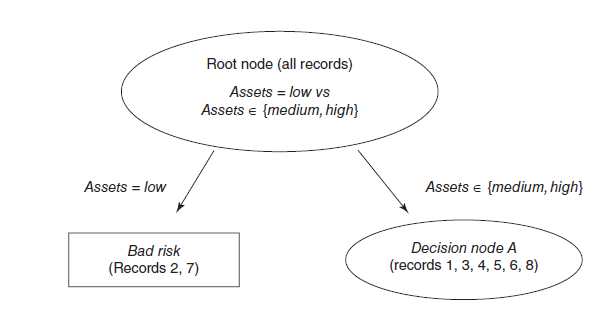
Първата стъпка, която трябва да се направи, е да се реши коя от трите променливи Savings, Assets и Income ще стои в корена на дървото. Нека представим набора от данни със всички възможни разделения:



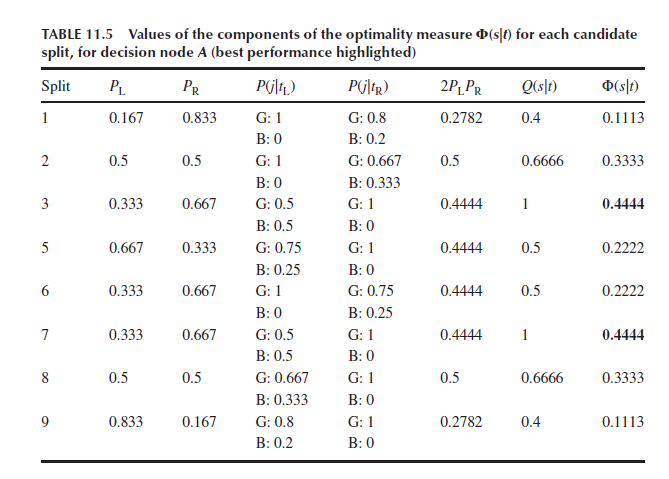
Получихме 9 възможни разделения. Използвайки формулите от метриката мярка за доброта можем да решим с коя от променливите можем да започнем дървото, т.е. коя от променливите ще стои в нашия корен и съответно кои стойности на нашата променлива ще лежат отляво и отдясно на корена. Изчислявайки функцията ще получим следните резултати:



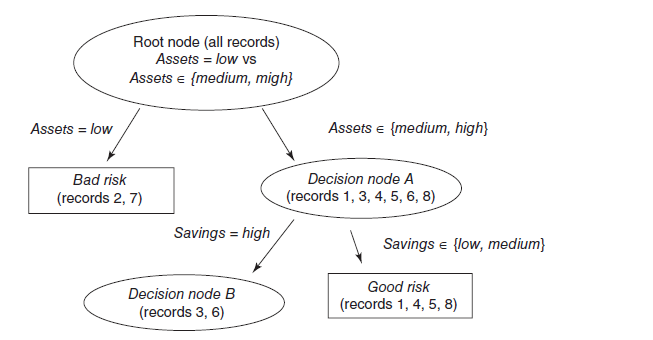
Най – голяма стойност за функцията получихме в четвъртия кандидат, т.е. променливата Assets ще стои в корена на дървото, отдясно на корена ще имаме Assets=low, а отляво на корена ще имаме Assets=medium/high. До тук дървото ще изглежда по следния начин:



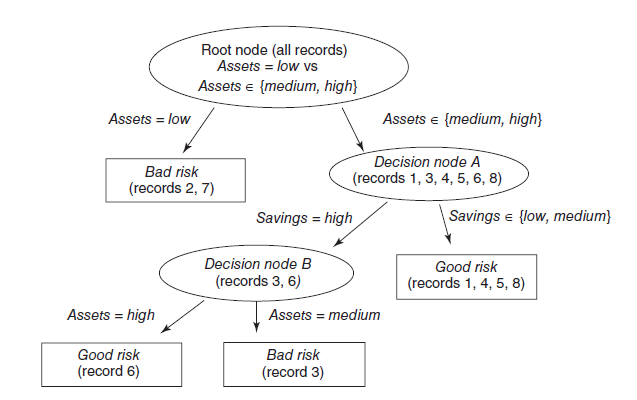
В лявата част на корена получаваме leaf node (листо) с резултат лош риск, защото всички записи в набора от данни с Assets=low са класифицирани с лош кредитен риск (bad risk), тоест кандидати 2 и 7. От дясната страна на дъврото остават кандидатите за разделяне 1,2,3,5,6,7,8,9 (всички, без избрания преди малко кандидат). Сега отново, използвайки формулата за изчисления на може да бъде взето решение как да се раздели новополучилия се десен node. Ето и изчисленията:

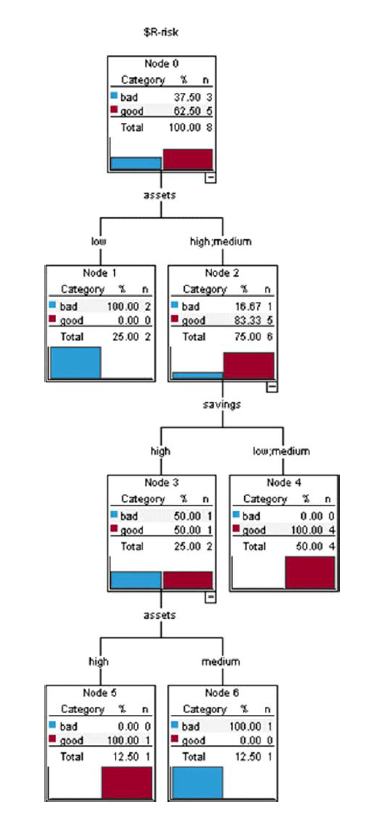


В случая получихме две равни най – големи стойности, а именно за кандидатите 3 и 7. В такъв случай алгоритъмът взима първия кандидат, за който е изчислена функцията, тоест кандидат 3. Ако се върнем малко по – нагоре до таблица 11.3 можем да видим, че избирайки кандидат 3, то в decision node A на дървото ще стои променливата Savings, левият branch ще бъде Savings=high, а десният Savings=low, medium. Дървото ще изглежда по следния начин:



От таблица 11.2 можем да видим, че комбинирайки Assets=medium/high и Savings=low/medium (кандидати 1,4,5,8) получаваме класификация само с добър кредитен риск, в дясно ще имаме leaf node с решение Good risk. Отляво остава да се вземе решение кой кандидат да се постави в Decision node B измежду останалите кандидати 3 и 6. Комбинирайки кандидатите 3 и 6, тоест Savings=High и Assets=Medium/High виждаме, че вземайки кандидат 3 получаваме лош кредитен риск, а за кандидат 6 – добър кредитен риск. Завършеното дърво ще изглежда така:





Ето някои приложения на CART методите:

* Сегментиране: Позволява автоматично разпределяне на наблюденията по групи, например разделяне на клиентите на фирма на групи по най-съществените им предпочитания.
* Класификация в степувани групи: Разделяне на наблюденията в категории чрез степенуване, напр. висока категория, средна, ниска
* Предсказване: Създаване на правила и използването им за предсказване на бъдещи събития, напр. доколко някой се очаква да изпадне в неплатежоспособност
* Намаляване на размерността на задачата (броя на променливите и/или случаите): Подбор на подмножество от предиктори като най-полезни за анализа и създаване на модел, описващ достатъчно добре цялия обем от данни.
* Разкриване на взаимодействия: Отделяне на специфични подгрупи в извадката, които си взаимодействат.
* Сливане на категории и дискретизиране на непрекъснати променливи: Преподреждане на групови предиктори и непрекъснати променливи с минимална загуба на информация

1. Алгоритъм C4.5

C4.5 e един от алгоритмите, използвани за генерирани на т.нар. дърво на решенията. Разработен е от Ross Quinlan и е разширение на неговия по-ранен алгоритъм ID3.

Алгоритъмът рекурсивно обхожда всеки възел и избира оптималното разделяне, докато това е възможно. За разлика от други алгоритми като CART, C4.5 не е ограничен до бинарни разделяния и съответно може да създава дървета с по-голям брой разклонения. Когато разглежданата променлива е категорийна, C4.5 по-подразбиране създава разклонение за всяка нейна стойност. Това не винаги е удачно, тъй като някои от стойностите може да се срещат по-рядко или пък са естествено свързани с други такива.

C4.5 използва концепцията **за информационна печалба** или **намаляване на ентропията**, за да се избере оптималното разделяне. Първо се изчислява най-малкият брой битове средно за символ, необходими за предаването на поток от символи, представящи стойностите на дадена променлива X. За целта се използва формулата:

е вероятността на срещане на j-тата стойност на X. Резултатът се нарича ентропия на X. Формулата се базира на факта, че за събитие с вероятност p средното количество информационни битове за предаване на резултата е . Например резултатът от хвърляне на монета (честна) с вероятност 0.5, може да се предаде, като се използва (0 или 1 в зависимост от резултата). Когато възможните изходи са повече от два, се търси претеглената сума на отделните логаритми, като теглата са равни на вероятностите за появата на съответните изходи.

Концепцията за ентропия се използва по следния начин. Да предположим, че разделяне S, дели тренировъчните данни T на няколко подмножества. Средното информационно изискване може да се изчисли като претеглената сума от ентропиите на отделните подмножества:

Тук e пропорцията на записите, попаднали в подмножество . По този начин информационната печалба може да се дефинира като , тоест увеличението на информацията в резултат от разделянето на тренировъчните данни T спрямо S. Във всеки възел, при който става разклонение, C4.5 избира разделянето S, което води до най-голямата информационна печалба .

Алгоритъмът C4.5 може да бъде представен чрез следната последователност от стъпки:

1. *Проверка дали са удовлетворени условията за край.*
2. *Изчисляване на информационната печалба за всички атрибути.*
3. *Избиране на атрибута, чието разделяне води до най-голямата печалба.*
4. *Създаване на decision node спрямо атрибута, избран в 3.*
5. *Разделяне на множеството от данни спрямо decision node-a от 4.*
6. *Повторно (рекурсивно) извикване на алгоритъма за всички подмножества, получени от разделянето в 5.*
7. *Прибавяне на новите разклонения на дървото, получено от 6. към decision node-а от 4.*
8. *Връщане на дървото.*

Примерен псевдокод:

