UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE

Rozhodovací stromy a jejich aplikace

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **prof. RNDr. Karel Hron, Ph.D.**
Vypracoval(a): **Maryna Korablova**
Studijní program: B1103 Aplikovaná matematika
Studijní obor Matematika–ekonomie se zaměřením na bankovnictví/pojišťovnictví
Forma studia: prezenční
Rok odevzdání: 2022

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Maryna Korablova

Název práce: Rozhodovací stromy a jejich aplikace

Typ práce: Bakalářská práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: prof. RNDr. Karel Hron, Ph.D.

Rok obhajoby práce: 2023

Abstrakt: Tato práce se věnuje problematice regresních a klasifikačních stromů, které patří pro svou relativní jednoduchou konstrukci a dobrou interpretovatelnost mezi stěžejní metody statistického učení. Po úvodu do problematiky se bakalářská práce věnuje souvisejícím specifickým tématům jako jsou prořezávání, náhodný les a bagging. Teoretické úvahy jsou doplněny a vlastní příklady, na kterých je demonstrována použitelnost regresních a klasifikačních stromů v praxi.

Klíčová slova: Regresní strom, klasifikační strom, prořezávání, křížová validace, bagging, náhodný les.

Počet stran: 52

Počet příloh: 1

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Maryna Korablova

Title: Decision trees and their applications

Type of thesis: Bachelor's

Department: Department of Mathematical Analysis and Application of Mathematics

Supervisor: prof. RNDr. Karel Hron, Ph.D.

The year of presentation: 2023

Abstract: This thesis is devoted to regression and classification trees, which belong to fundamental methods of statistical learning because of their relatively simple construction and good interpretability. After introduction to the topic the bachelor thesis deals with related specific topics like pruning, random forest and bagging. Theoretic considerations are accompanied by examples where applicability of regression and classification trees in practice is demonstrated.

Key words: Regression tree, classification tree, tree pruning, cross validation, bagging, random forest.

Number of pages: 52

Number of appendices: 1

Language: Czech

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně pod vedením pana prof. RNDr. Karla Hrona, Ph.D. a všechny použité zdroje jsem uvedla v seznamu literatury.

V Olomouci dne ....

podpis

Obsah

Poděkování

Ráda bych poděkovala svému vedoucímu bakalářské práce panu prof. RNDr. Karlu Hronovi, Ph.D. za trpělivost a užitečné rady. Za podporu a důvěru v mou osobu. A taky bych chtěla poděkovat své spolubydlící Nikol Krupičkové za korekturu českého jazyka a Veronice Šmajserové.

Úvod

Po chvilce přemýšlení jsem se rozhodla vybrat si téma týkající se rozhodovacích stromů a jejich aplikaci v praxi. Vzhledem k tomu, že bych svůj budoucí profesní život chtěla spojit s analýzou dat, zjistila jsem, že toto téma je velmi zajímavé. Rozhodovací stromy nevyžadují přísné předpoklady na analyzovaná data a přitom získáme poměrně dobře interpretovatelné výsledky v regresním, resp. klasifikačním kontextu.

V rámci této práce bych chtěla nejprve představit základních pojmy, jako jsou obecné principy rozhodovacích stromů, rozdělení pozorování do konkrétních skupin, návrhy stromů, a též metody, které se používají ve stromech ke zjednodušení interpretace a současně zlepšení predikční schopnosti při řešení konkrétních problémů. Mým druhým cílem je pak právě aplikovat získané znalosti v praxi. Budu konstruovat stromy v programu R, vizualizovat a snažit se podrobně vysvětlit, co se děje v konkrétních krocích.

Kapitola 1

Rozhodovací stromy a jejich obecné principy

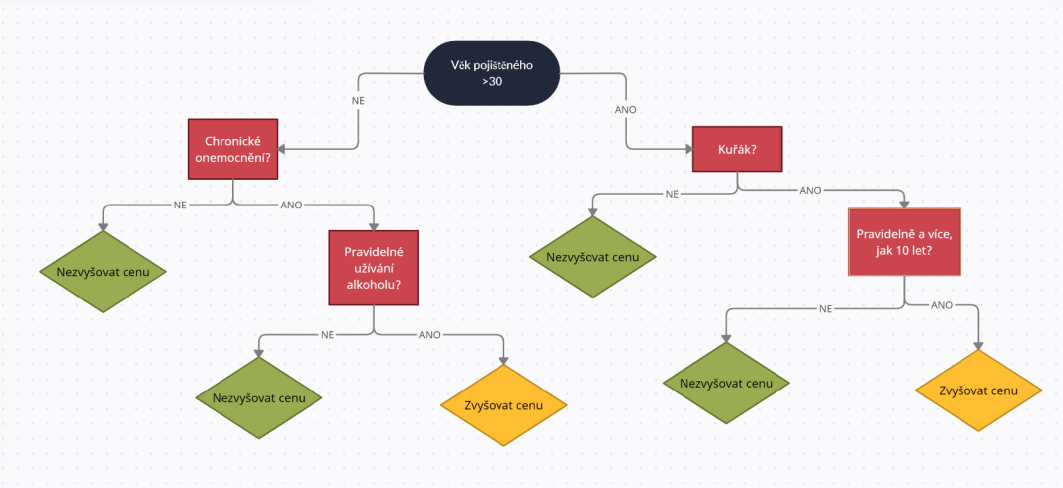
V této kapitole jsem použila tuto literaturu: [3], [8].

1.1. Základní pojmy a zásady

Rozhodovací stromy jsou jedny z nejoblíbenějších a nejstarších principů klasického strojového učení. Simulují procesy, které lidé používali bez pomoci programů a algoritmů. V rámci této práce však nedojde k pohledu na téma z hlediska strojového učení, ale ke statistické analýze dat, ve které rozhodovací stromy zaujímají určitou užitečnou roli.

Například zaměstnanci bank nebo pojišťoven by se setkali s klienty a přímo by jim položili určité otázky, které by později mohly ovlivnit to, zda klient dostane půjčku nebo kolik musí člověk zaplatit při podepsání pojistné smlouvy. Zde uvedu vizuální příklad ceny pojistného ze strany vnímání pojišťovny.

Podíváme se na obrázek číslo 1.1. Aby pojišťovna přidělila spravedlivou částku pojištění, zvažuje určité aspekty života pojištěného. Dívá se například na to, jak je člověk starý. Pokud je mu přes třicet let, pak se zeptá, zda dotyčný kouří. Pokud je odpovědí ne, pak se rozhodne nezvyšovat pojistnou částku, ale pokud je odpověď ano, položí další otázku. Pokud člověk kouří vice jak deset let, tak se rozhodne zvýšit pojistnou částku, pokud ne, tak to nechá. Totéž se stane, pokud půjde po jiné větvi, a to: pokud je člověku pod třicet let. Pojišťovna se podívá do lékařské



Obrázek 1.1: *Jednoduchý příklad fungování stromu na intuitivních problémech.*

dokumentace a zjistí, že má člověk nějaké vážné chronické onemocnění, a poté se pojistnou částku rozhodne zvýšit, jinak je to pro pojišťovnu příliš velké riziko. Samozřejmě ve skutečnosti je tento proces velmi odlišný. Existuje mnohem více proměnných. Ale pro jasný příklad fungování algoritmu je toto schéma vhodné.

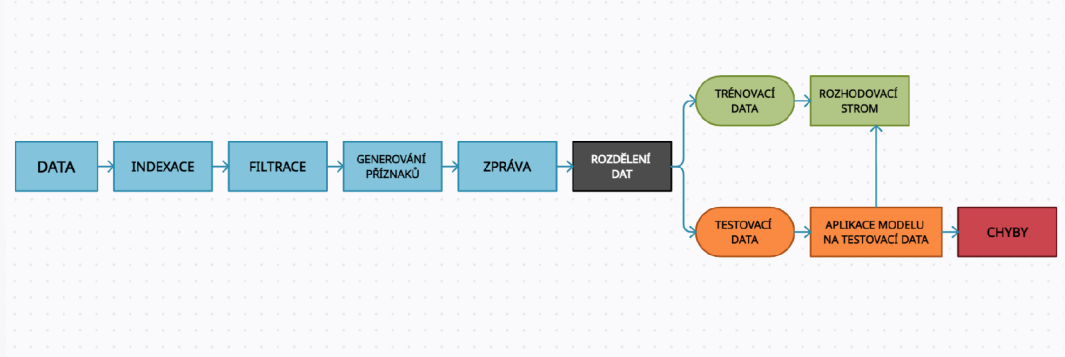
Systém pokaždé převezme jeden z příznaků, rozdělí data do dvou tříd a snaží se zajistit, aby byl maximalizován počet „ano“ a počet „ne“ ve výsledných podmnožinách dat. Samotný stroj si vždy vybere jeden z příznaků a podle něj rozdělí dostupná data, potom nakonec dospěje k rozhodnutí.

Rozhodnutí se provádí pokaždé s určitým stupněm pravděpodobnosti, a čím vyšší je tato pravděpodobnost, tím lepší a přesnější bude výsledek analýzy rozhodovacího stromu. To znamená, že nakonec dostaneme soubor logických pravidel. Pokud je hodnota znaménka A větší než určitá prahová hodnota „x“, a hodnota znaménka B je větší než určitá prahová hodnota „y“, pak se rozhodneme. Například: „zvýšit cenu pojistné částky“. To je základní princip algoritmů rozhodovacího stromu.

Proč se rozhodovací stromy staly tak populárními ve statistice a analýze dat navzdory skutečnosti, že se jedná o jeden z nejstarších algoritmů? Odpověď na tuto otázku spočívá ve velmi podstatných výhodách. Za prvé, je to jednoduchost v porozumění, jasnosti a interpretaci. Na rozdíl například od hlubších

technik strojového učení, jako jsou neuronové sítě. Za druhé, rozhodovací stromy odvádějí velmi dobrou práci i s velkým datovým souborem. Za třetí, rozhodovací stromy mohou pracovat s kategoriemi i kvantitativními hodnotami.

Pokud hovoříme o výhodách, musíte specifikovat také nevýhody, které jsou v tomto modelu přítomny. Rozhodovací stromy jsou nestabilní. To se projevuje, pokud nějak chceme změnit data v existujícím stromu (nebo dochází k přeučování modelu), pak je často možné, že původní malá změna na vstupu povede k velké změně modelu i výstupních hodnot. Například první věc, na kterou se podíváme, není věk osoby při rozhodovaní o pojistné částce, ale chronické onemocnění. Přesněji řečeno, při přeškolení se změní kořenový uzel stromu. A druhou nevýhodou je, že je obtížné kontrolovat velikost stromu, což často vede ke zbytečným větvím. Avšak v tomto případě existuje řada algoritmů, které mohou pomoct i s řešením tohoto problému. Tyto algoritmy uvedu později.



Obrázek 1.2: *Podrobná schéma probíhání analýzu uvnitř systému.*

Na dalším obrázku (obrázek 1.2) chci uvést jasný příklad toho, jak vypadá optimalizace modelu. Zde vidíme, že po úvodním předzpracování dat a obdržení zprávy o vygenerovaných příznacích jsou data rozdělena na testovací a trénovací. Trénovací data jsou 70% - 80% procent, pak je model sestaven a použije se na zbývajících 20% - 30%, které nebyly použity v procesu učení. A pokud se chyba, kterou testovací model dělá, příliš neliší od chyby, kterou dělá na trénovacích datech, pak říkáme, že model je stabilní a funkční, a lze jej prakticky použít. Pokud se objeví, že chyb je více, pak je to signál, že došlo k přeškolení modelu,

že bylo použito příliš mnoho specifických příznaků a model nemá prediktivní sílu.

Můžete například použít nějakou pojišťovnu, která vede následující prognózu k riziku výskytu konkrétní pojistné události. Za příznak vezme rodné číslo člověka a bude tvrdit, že lidé s takovými čísly mají větší riziko chronických onemocnění. V tomto případě bude model naprosto bezvýznamný, protože, jak je známo, rodné číslo osoby u ní nemůže ovlivnit výskyt chronických onemocnění. Toto je příklad přeškolení modelu, když systém používá příliš specifické příznaky. Ověření modelu na testovacích datech vylučuje takové přeškolení. Metody, které vylučují přeškolení, jsou často integrovány do samotného modelového systému. Toto je podstatou tzv. křížové validace. Když samotný systém ukládá určité množství dat po celou dobu, učí se z trénovacích dat a pak je zkontroluje na testovacích datech. Role trénovacích, resp. testovacích dat se navíc pro konkrétní data v průběhu postupu křížové validace mění.

1.1.1. Terminologie

V této podkapitole chci představit terminologii, kterou v budu budoucnu aktivně používat pro podrobný a strukturovaný popis tohoto problému.

Rozhodovací strom – je souvislý graf neobsahující cykly s orientovanými hranami, který se skládá z kořene, větví (hran), terminálních a neterminálních uzlů.

Rozhodovací pravidlo – je to pravidlo typu „Jestliže..., potom...“, které rozhoduje, jestli nějaký objekt patří k dané třídě.

Uzel – v takzvaných uzlech existují rozhodující pravidla a kontrola dat se také provádí na základě hodnoty trénovacích dat (pokud splňují podmínku, která je v uzlu napsána). Následně jsou trénovací data dle tohoto uzlu rozdělena. Pak je rozděleno do dvou kategorií: na skupinu, která splňuje pravidlo a skupinu, která nesplňuje pravidlo. Poté se kontrola použije znovu na nově vytvořený uzel (proces se opakuje iterativně za účelem optimalizace daného kritéria).

List – koncový uzel stromu. List obsahuje hodnoty cílové funkce.

Kořenový uzel – je počáteční uzel stromu, ze kterého začíná samotný proces

konstrukce. Kořen je ten vrchol, do nějž nevstupují žádné větve a z něhož vstupuje do ostatních uzlů vždy jedna větev.

Trénovací data – podmnožina dat, která je užita pro učení analytických modelů.

Testovací data – jsou to data, pomocí kterých je program testován. Potvrzuje správnost zadaných dat a fungování algoritmu.

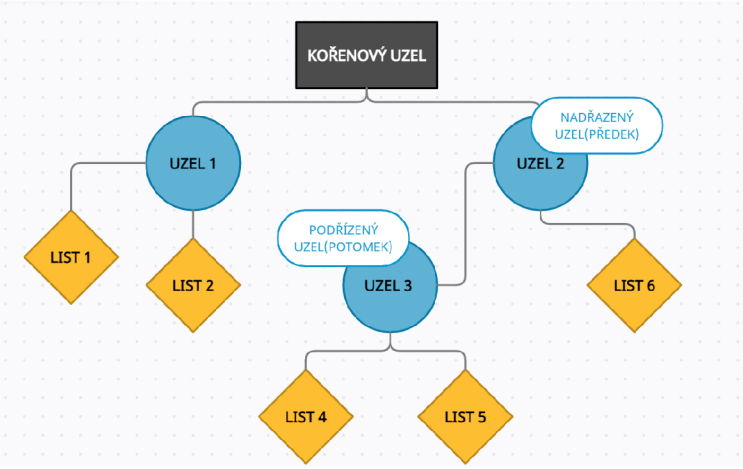
1.2. Struktura rozhodovacích stromů

Struktura rozhodovacích stromů je uspořádána jako hierarchická posloupnost otázek typů „Pokud..., pak...“. Pravidla jsou generována na základě trénovacích dat. Tato struktura by měla zahrnovat prvky dvou typů – uzly a listy. Pokud mluvíme o uzlech, pak, jak bylo řečeno v terminologii, zahrnují pravidla, podle kterých probíhá kontrola pozorování. Jejich hlavním úkolem je najít shodu pozorování s vybraným atributem trénovací množiny.

Hlavní rozdíl mezi listem a uzlem spočívá v tom, že list neobsahuje podmínky, ale pouze podmnožinu objektů, která splňuje všechna pravidla větve, která končí na tomto listu. Uzel je deklarován jako list, pokud byla splněná určitá podmínka pro zastavení algoritmu. Pozorování se může dostat na list pouze v případě, že na cestě k němu splnilo všechna pravidla. Jedinečnost řešení je zajištěna skutečností, že existuje pouze jedna cesta k listu a pozorování se může dostat pouze do jednoho listu.

Na tomto diagramu (obrázek 1.3) vidíme, jak vypadá samotný rozhodovací strom. Za prvé, máme kořenový uzel, ze kterého začíná proces konstrukce. Poté z něj vychází dvě větve, které vedou k Uzlu 1 a Uzlu 2, které se nazývají také nadřazenými uzly nebo „předky“. Pojďme se blíže podívat na Uzel 2. Jak je známo, obsahuje rozhodující pravidlo, na jehož základě dochází ke kontrole dat. Můžeme pozorovat, jak na základě testu systém rozdělil tento Uzel 2 na List 3 a Uzel 3.

Jak jsem popsala dříve, uzel je deklarován jako list, pokud byla zapojena nějaká podmínka zastavení, což znamená, že se tímto způsobem objevil List 3. V Uzlu 3 nebyla tato podmínka naplněna, takže byl rozdělen do dalších dvou



Obrázek 1.3: *Pojmenování jednotlivých „bloků“ v stromech.*

uzlů, které byly později deklarovány jako listy (List 4 a List 5).

1.3. Typy rozhodovacích stromů

Rozhodovací stromy jsou rozděleny do dvou typů, a to jsou regresní a klasifikační, kde druhý typ je speciálním případem prvního. Podrobná analýza těchto dvou typů bude probíhat v samotných kapitolách. Proč existuje rozdělení na tyto dva typy? Odpověď na tuto otázku spočívá v datech, se kterými musíme pracovat. Regresní stromy pracují se kvantitativními proměnnými a klasifikační stromy s kvalitativními proměnnými v roli závisle proměnné. Hodnoty u listů potom v regresních stromech odpovídají průměru závisle proměnné pro pozorování odpovídající dané cestě a v klasifikačním případě potom analogicky nejčastěji se vyskytující třídě (kategorii) pozorování.

1.3.1. Základní metody a algoritmy

V úvodu se nebudu ponořovat do principu těchto algoritmů, ale pouze je vyjmenuji a uvedu klady a zápory. V následujících částech budu podrobně analyzovat pouze metodou CART, jelikož je častěji použitelná v praxi.

Metoda CART (Classification And Regression Tree)

Začnu snad jedním z nejstarších algoritmů, který je určen jak pro regresní stromy, tak i pro klasifikační stromy. Byl navržen v roce 1983 a podíleli se na něm následující vědci: Leo Breiman, Jerome Friedman, Richard Olshen a Charles J. Stone [1].

Algoritmus konstruuje binární rozhodovací stromy, které obsahují jenom dva potomky v každém uzlu. V procesu práce se pozorování z trénovacích dat rekurzivně rozdělují na podmnožiny. Tato podmnožina obsahuje objekty, které mají stejnou hodnotu cílové proměnné.

Výhody tohoto algoritmu: není založený na žádných stochastických předpokladech, například o rozdělení pravděpodobnosti vstupních dat. Atributy rozdělování vybíráme během procesu konstrukce stromu, a proto není nutné předem aplikovat metody pro výběr proměnných. Je stabilní vůči odlehlým hodnotám a výpočetně efektivní. K nevýhodám lze přičíst nestabilitu z hlediska dat: dokonce i malé úpravy v testovacích datech vyvolávají významné změny ve struktuře rozhodovacího stromu.

Metoda ID3 (Iterative Dichotomizer 3) a C4.5

Algoritmus, který je určen pouze k řešení problémů s kvalitativními (závisle) proměnnými, to znamená, že je metoda, která je vhodná pouze pro klasifikační stromy. Byl vyvinut Johnem R. Quinlanem. Počet potomků v uzlu stromu není omezeno. Nemůže však pracovat s chybějícími hodnotami, takže před použitím tohoto algoritmu je nutno zkontrolovat přítomnost chybějících hodnot.

Základní myšlenkou je, že trénovací třída podléhá rekurzivnímu dělení do podmnožin pomocí pravidel řešení, kde se tyto třídy nachází v kořenovém uzlu rozhodovacího stromu. Hlavní myšlenka algoritmu spočívá v postupném dělení trénovacího datového souboru pomocí rozhodovacích pravidel. Rozdělení pokračuje, dokud výsledné podmnožiny neobsahují pozorování pouze jedné třídy, poté se proces učení zastaví a podmnožiny jsou deklarovány jako listy stromu, které obsahují řešení.

Následně byl tento algoritmus vylepšen, jde o C4.5. Jeho hlavním rozdílem od svého předchůdce bylo to, že mu byla přidána schopnost pracovat s chybějícími hodnotami atributů. Tento algoritmus jsem uvedla jen pro zajímavost, jelikož se v analýze dat nejčastější používá metoda CART.

Kapitola 2

Klasifikační stromy

Pro vysvětlení tématu jsem použila zdroje [3], [6]. V této kapitole budu podrobně analyzovat základní metody konstrukce klasifikačních rozhodovacích stromů, a taky princip jejich fungování.

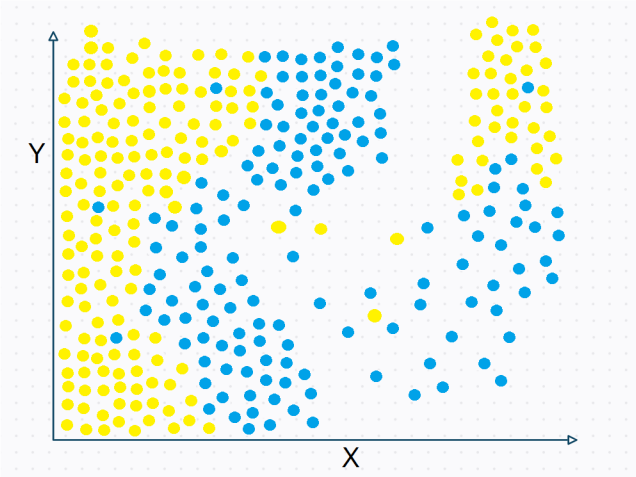
Klasifikační stromy samotné nedávají z hlediska klasifikace moc dobré výsledky. Ale proč je tedy používáme tak často? Je tomu proto, že využíváme nějakou kombinaci klasifikačních stromů. Jinými slovy, snažíme se vytvářet lepší metody a algoritmy ze slabších klasifikátorů. Jak jsem již uvedla, klasifikační stromy mají velmi důležitou výhodu, a to je interpretovatelnost. Když sestavíme strom, můžeme snadno pochopit, jak funguje samotný proces analýzy dat. Na základě získaných dat se naučíme vytvářet složitější modely, které budou mít v budoucnu méně chyb a větší predikční schopnosti.

Máme například marketingovou firmu, která se chce naučit, jak nejlépe propagovat svůj produkt tak, aby si jej zákazník koupil. Tato firma se obrací na pomoc analytiků, kteří budou následně nuceni prezentovat vykonanou práci zaměstnavateli. Nejlepší je v tomto případě použít právě klasifikační stromy. Analytik provede analýzu, identifikuje věkovou kategorii produktu a další vysvětlující proměnné pro zařazení do jedné z kategorií koupil – nekoupil a na základě toho se podívá na to, kam je lepší umístit reklamu a v kolik hodin. Po odvedené práci analytik představí výsledky zaměstnavateli, pro kterého bude snažší pochopit princip stromů, na rozdíl od jiných složitějších metod klasifikace.

2.1. Metody konstrukce klasifikačních stromů

Nejprve musíme zdůraznit, že když mluvíme o klasifikačních stromech, obvykle používáme metodu CART. To je základní algoritmus, pomocí kterého konstruujeme strom.

Uvedu jednoduchý příklad jedné z variant popisu konstrukce stromů.



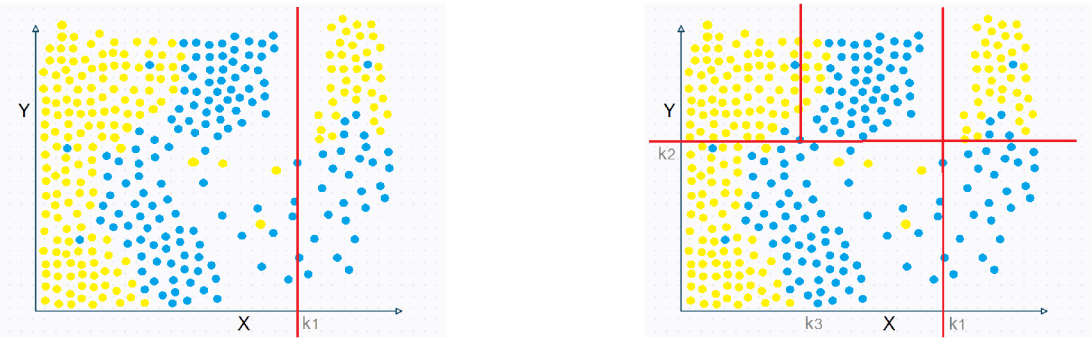
Obrázek 2.1: *Prostor obsahující dvě třídy pozorování.*

Na obrázku 2.1 máme dvě třídy s kroužky dvou barev - žluté a modré. V tomto příkladu jsou dva prediktory: „X“ a „Y“. Pak probíhá rozdělení roviny přímkou, která je kolmá k jedné z os souřadnic tak, aby ta přímka oddělila třídu *„žlutých kroužků“* od třídy *„modrých kroužků“* (dělení probíhá pouze kolmými přímkami! V prostoru R^n by se jednalo o nadroviny, které jsou rovnoběžné s osami souřadnic).

Naším úkolem je oddělit třídu s žlutými kroužky tak, aby skoro neobsahovala žádná pozorování z třídy modrých kroužků. Na obrázku 2.2 budeme postupně oddělovat přímkami k_1, k_2 a k_3 tyto třídy.

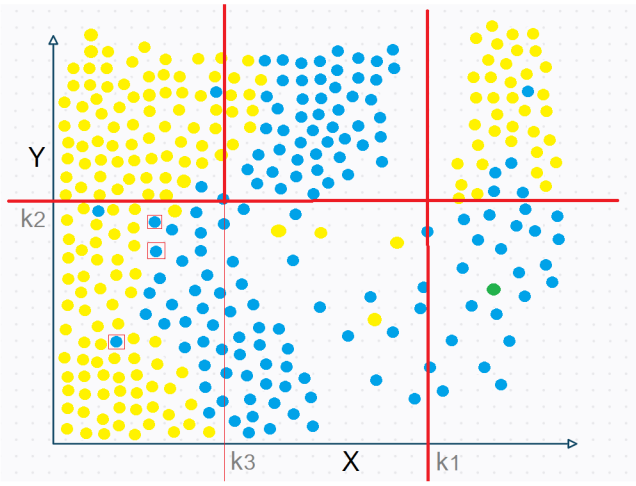
Dělení „může probíhat“, dokud nebudou všechny prvky třídy *„žluté“* odděleny od třídy *„modré“*. Tak proč se to neděje v praxi? Odpověď na to bude popsaná v následujícím odstavci.

Předpokládejme, že oddělíme několik zbývajících prvků třídy *„modré“*, jak je



a) b)

Obrázek 2.2: *Rozdělení prostoru na jednotlivé podoblasti podle určité třídy.*



Obrázek 2.3: *Finální podoba rozdělení pozorování na jednotlivé podoblasti přímkami.*

zobrazeno na obrázku 2.3. Jakmile toto rozdělení vznikne, analytik jej bude chtít v budoucnu aplikovat na nová data. Předpokládejme, že v některých oddělených datech je nové pozorování - *„zelený kroužek“*. Analytik jej logicky přiřadí k převládající barvě dané třídy. Ale co stane, když toto pozorování vznikne v „obdélníku“, ve kterém je pouze jedna značka třídy *„modré“* a analytik přiřadí pozorování *„zelený kroužek“* k *„modré“*? V důsledku toho dojde k tzv. přeučení (overfitting) (1.1), které bylo zmíněno v první kapitole. V tomto případě probíhá rozhodování, které je založeno pouze na jednom bodě. Mluvit o nějakých důsledcích, když se opíráme pouze o jedno pozorování, je nespolehlivé.

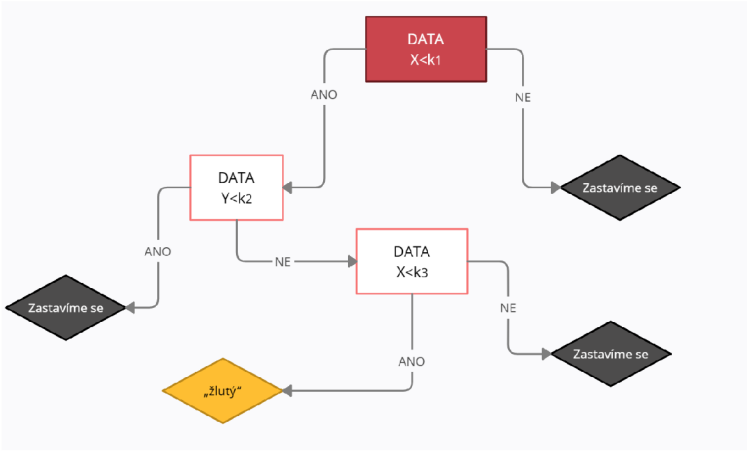
Druhou možností, jak popsat klasifikační stromy, je sada logických pravidel.

Například můžeme popsat obdélník, který vznikl po rozdělení roviny přímkami k_1, k_2 a k_3 takto. Nechť

$$(X < k_1) \wedge (Y > k_2) \wedge (X < k_3),$$

pak nové pozorování zařadíme do třídy *„žluté“*.

Následně přichází i poslední pohled na klasifikační stromy, který vlastně vysvětluje pojem „strom“. Celý výše popsaný proces může být totiž zobrazen graficky, viz obrázek 2.4.



Obrázek 2.4: *Grafické znázornění procesu konstrukce stromu.*

V jednotlivých obdélnících jsou pozorování. Kořenový uzel se rozdělil na dva následující obdélníky. K větvím se přidávají podmínky, v tomto příkladu je to nejprve podmínka, že $(X < k_1)$. Což znamená, že na levé straně jsou všechna data, která splnila uvedenou podmínku, a na pravé straně ta, která ji nesplnila. A tak dál, dokud se algoritmus nezastaví.

2.1.1. Metoda CART

Pojem „čistá“ data

Na základě předchozího příkladu je měřítkem čistoty dat to, kolik prvků tříd, které se k němu vztahují, jsou v jednotlivých uzlech. K určení míry čistoty dat se používá několik metod.

1) Giniho index

Pokud datová sada T obsahuje data n tříd, pak je Giniho index definován pro pozorování z dané podoblasti, vzniklé dělením v rámci klasifikačního stromu, jako:

$$Gini(T) = \sum_{j=1}^{n} p_j(1 - p_j), \tag{2.1}$$

kde p_j je část trénovacích pozorování z třídy j. Čím je hodnota Giniho indexu menší, tím vyšší máme jistotu, že uzel obsahuje pouze jednu třídu.

2) Entropie je další používanou charakteristikou

$$H = -\sum_{j=1} p_j \cdot \log p_j, \tag{2.2}$$

kde $0 \leq p_j \leq 1$, a pak $0 \leq -p_j \cdot \log p_j$. Pokud máme „čistá“ data, pak entropie bude nabývat hodnotu blízkou nule.

2.2. Příklady klasifikačních stromů

V této kapitole bych chtěla představit možnosti praktického použití klasifikačních stromů v programovacím prostředí R, a to konkrétně v R Studiu. Budu využívat matematický aparát, který jsem popsala v předchozí kapitole o metodě CART. Pokusím se vysvětlit každý krok a popsat následující grafy a schéma.

Za tímto účelem jsem vybrala data, které jsou spojená s mozkovou mrtvicí [2]. Než přistoupíme k analýze dat pomocí metody rozhodovacích stromů, je nutné připravit data pro analýzu. Při psaní kódu vám podrobněji vysvětlím obsah tohoto datového souboru, a jak ve skutečnosti v R probíhá samotný výpočet.

V první řadě připojuji potřebné knihovny ke svému projektu. Funkce *set.seed* generuje náhodné hodnoty, které jsou jedinečné pro *seed* (a budou stejné bez ohledu na počítač, na kterém pracujete, a proto zajistí reprodukovatelnost).

Postupně, ale jistě přichází okamžik, kdy se odehrává samotné kouzlo rozhodovacích stromů. Vytvořila jsem proměnnou *BrainStrokeData*, která obsahuje in-

formace o datech z excelovské tabulky. Potom jsem přidala kód *BrainStrokeData = na.omit(BrainStrokeData)*, který vlastně vymaže všechny chybějící hodnoty, které pak můžou způsobit chyby. Taky v původních datech byla proměnná „id“, jelikož nemá predikční sílu, vymazala jsem ji. K prohlížení obsahu našich dat používáme funkci *View(BrainStrokeData)*. Soudě podle tabulky níže vidíme následující situaci: máme jedenáct sloupců gender(pohlaví), age (věk) hypertension (hypertenze) heart disease (srdeční choroba), ever married (jestli osoba byla někdy v manželství), work type (typ prací), Residence type (typ bydliště), avg glucose level(průměrná úroveň glukózy), bmi (index tělesné hmotnosti), smoking status (kuřák/nekuřák), stroke (mrtvice). Pro jednoduší interpretace jsem změnila nuly a jedničky na ANO/NE.

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
15	Male	75.00	ANO	NE	Yes	Private	Urban	221.29	25.8	smokes	ANO
470	Female	28.00	NE	NE	Yes	Private	Urban	183.45	40.5	smokes	NE
47	Male	73.00	ANO	NE	Yes	Self-employed	Urban	194.99	32.8	never smoked	ANO
283	Female	22.00	NE	NE	No	Private	Urban	69.94	22.8	Unknown	NE
303	Male	64.00	NE	NE	Yes	Govt_job	Urban	239.64	34.6	formerly smoked	NE
72	Male	47.00	NE	NE	Yes	Private	Urban	86.94	41.1	formerly smoked	ANO
31	Male	74.00	NE	NE	Yes	Private	Rural	219.72	33.7	formerly smoked	ANO
100	Female	38.00	NE	NE	No	Self-employed	Urban	82.28	24.0	formerly smoked	ANO
258	Male	58.00	ANO	NE	Yes	Govt_job	Rural	56.96	26.8	smokes	NE
330	Female	18.00	NE	NE	No	Private	Urban	80.05	24.2	never smoked	NE
271	Female	43.00	NE	NE	Yes	Self-employed	Rural	115.22	21.2	Unknown	NE
113	Female	71.00	NE	NE	Yes	Govt_job	Urban	263.32	38.7	never smoked	ANO
319	Female	29.00	NE	NE	Yes	Private	Urban	63.69	28.1	smokes	NE
361	Female	22.00	NE	NE	No	Private	Rural	107.52	41.6	Unknown	NE
440	Female	31.00	NE	NE	Yes	Private	Urban	98.99	31.2	never smoked	NE
17	Female	71.00	NE	NE	Yes	Govt_job	Rural	193.94	22.4	smokes	ANO
29	Male	48.00	NE	NE	No	Govt_job	Urban	84.20	29.7	never smoked	ANO
438	Male	9.00	NE	NE	No	children	Rural	94.59	20.0	Unknown	NE
345	Male	70.00	ANO	NE	Yes	Self-employed	Urban	251.60	27.1	never smoked	NE
476	Male	67.00	NE	NE	Yes	Self-employed	Urban	68.52	26.2	never smoked	NE
418	Female	7.00	NE	NE	No	children	Rural	72.35	17.0	Unknown	NE
75	Male	81.00	NE	NE	Yes	Private	Urban	72.81	26.3	never smoked	ANO
42	Female	39.00	ANO	NE	Yes	Private	Rural	58.09	39.2	smokes	ANO
458	Male	29.00	NE	NE	Yes	Self-employed	Urban	118.70	33.2	Unknown	NE
311	Female	47.00	NE	NE	Yes	Private	Urban	136.80	37.3	never smoked	NE
354	Female	17.00	NE	NE	No	Private	Urban	87.52	39.2	never smoked	NE
396	Male	53.00	NE	NE	Yes	Private	Rural	116.66	28.5	formerly smoked	NE
469	Male	41.00	NE	NE	No	Govt_job	Rural	74.81	39.7	smokes	NE
390	Male	31.00	NE	NE	Yes	Govt_job	Rural	91.65	24.6	formerly smoked	NE
164	Female	74.00	NE	NE	Yes	Self-employed	Urban	74.96	26.6	never smoked	ANO
280	Female	29.00	NE	NE	No	Private	Urban	71.89	27.6	never smoked	NE
412	Female	42.00	NE	NE	Yes	Private	Rural	112.06	38.2	never smoked	NE
112	Female	68.00	ANO	NE	Yes	Self-employed	Rural	206.09	26.7	never smoked	ANO

Obrázek 2.5: *Tabulka, obsahující proměnné a pozorování z datového souboru „BrainStroke“.*

Dále pomocí *head(BrainStrokeData)* a *tail(BrainStrokeData)* chci vyhodnotit stav dat a zobrazit řádky. První věc, které si všimneme, je to, že data jsou tříděna podle hodnot proměnných, což není dobré pro naši analýzu. Proč tomu

tak je? Protože když bude následně potřeba rozdělit data na trénovací a testovací, na konci to dopadne tak, že program vybral deset procent z jedné skupiny a devadesát z druhé, což už říká o tom, že analýza nebude ani skoro přesná.

```
> head(BrainStrokeData)
# A tibble: 6 x 11
  gender   age hypertension heart_disease ever_married work_type     Residence_type avg_glucose_level   bmi smoking_status  stroke
  <chr>  <dbl> <chr>        <chr>         <chr>        <chr>         <chr>                     <dbl> <dbl> <chr>           <chr>
1 Male      67 NE           ANO           Yes          Private       Urban                     229.   36.6 formerly smoked ANO
2 Male      80 NE           ANO           Yes          Private       Rural                     106.   32.5 never smoked    ANO
3 Female    49 NE           NE            Yes          Private       Urban                     171.   34.4 smokes          ANO
4 Female    79 ANO          NE            Yes          Self-employed Rural                     174.   24   never smoked    ANO
5 Male      81 NE           NE            Yes          Private       Urban                     186.   29   formerly smoked ANO
6 Male      74 ANO          ANO           Yes          Private       Rural                      70.1  27.4 never smoked    ANO
> tail(BrainStrokeData)
# A tibble: 6 x 11
  gender   age hypertension heart_disease ever_married work_type    Residence_type avg_glucose_level   bmi smoking_status stroke
  <chr>  <dbl> <chr>        <chr>         <chr>        <chr>        <chr>                     <dbl> <dbl> <chr>          <chr>
1 Female    16 NE           NE            No           Never_worked Urban                     102.   27.1 never smoked   NE
2 Female    18 NE           NE            No           Never_worked Urban                      81.7  21.6 never smoked   NE
3 Male      13 NE           NE            No           Never_worked Urban                      85.1  14.6 Unknown        NE
4 Female    17 NE           NE            No           Never_worked Urban                      78.1  44.9 never smoked   NE
5 Female    18 NE           NE            No           Never_worked Urban                      97.6  21.5 Unknown        NE
6 Female    16 NE           NE            No           Never_worked Rural                      68.3  20.4 never smoked   NE
```

Obrázek 2.6: *Tabulka, obsahující proměnné a pozorování z datového souboru „BrainStroke“.*

K překonání tohoto problému používáme funkci *sample(1:nrow(BrainStrokeData))*, čímž vytvoříme novou proměnnou *new_ data*, která nyní bude obsahovat smíšené hodnoty.

Pak následuje funkce, která rozděluje data na trénovací a testovací. Vytvoříme proměnnou *trenovaci_ testovaci* a přidáme do ní pozorování podle pravidla 80/20. Postupujeme přitom tak, že pro datový soubor, v němž jsme předtím náhodně promíchali pořadí pozorování, vybereme do trénovací množiny prvních 80 procent pozorování.

Pak jsem se rozhodla zkontrolovat svou funkci[1] a rozměr dat.

```
> dim(trenovaci)
[1] 401  11
> dim(testovaci)
[1] 101  11
```

Skončili jsme s 11 proměnnými, 401 trénovacími a 101 testovacími pozorováními.

Funkce *prop.table()* se používá ve spojení s *table()* k ověření nahodilosti dat.

[1] Spočítáme počet řádků v datové množině *n_ radek = nrow(data)*, načež vrátíme *n*-tý řádek k vytvoření trénovací množiny. Poté vybereme z prvního řádku po *n*-tý *trenovaci_ sestavene = 1: t_ radek*. A pak to už je jednoduché - pokud je podmínka pravdivá, funkce vrátí trénovací množinu, v opačném případě - testovací množinu.

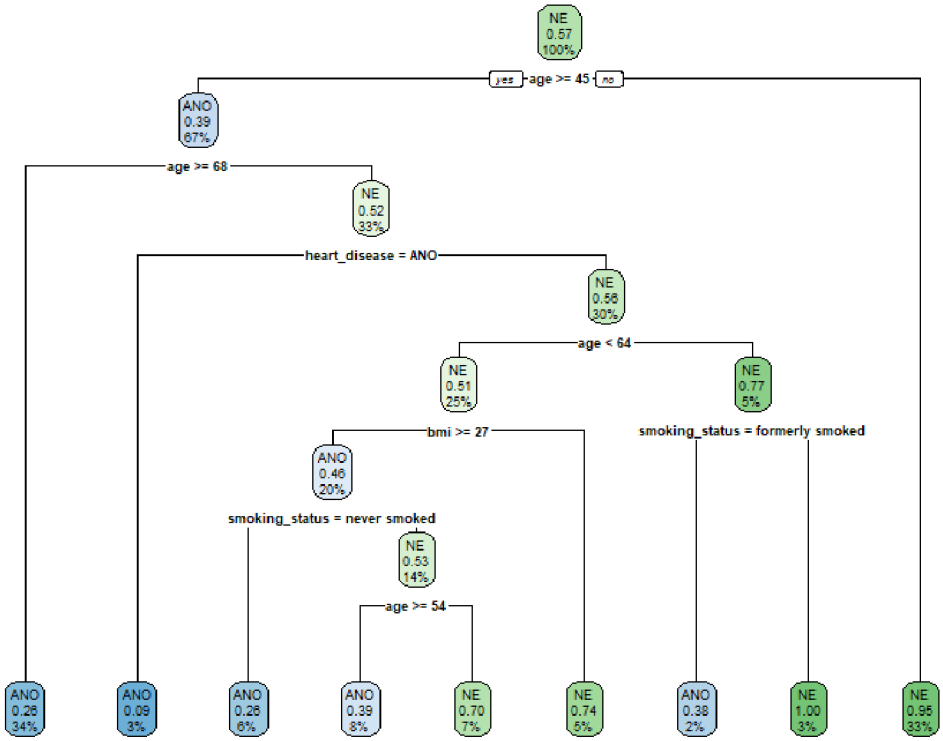
```
> prop.table(table(trenovaci$stroke))

      ANO        NE
0.4264339 0.5735661
```

Jak můžeme pozorovat, proporce jednotlivých kategorií se od sebe výrazně neliší. To dobře odpovídá počtu nemocných (209) a zdravých (300) v původní populaci.

Dalším krokem již začíná proces konstrukce rozhodovacích stromů pomocí funkce *rpart()*. Zde jsou argumenty: *trenovaci$stroke* specifikuje (kategoriální) závisle proměnnou a vysvětlující proměnné (v našem případě deset), *data = trenovaci* je množina dat, *metoda = 'class'* je metoda klasifikace, kterou používáme.

Použijme *rpart.plot(strom)* ke grafickému znázornění našeho stromu, abychom se podívali na výsledek a pochopili, co se tam děje.



Obrázek 2.7: *Výsledný klasifikační strom.*

Na výstupu kódu dostaneme výsledný klasifikační strom. Vidíme první kořenový uzel, který ukazuje celkovou pravděpodobnost mozková mrtvice. Je hned vidět, že se první uzel našeho stromu ptá program pomocí proměnné *age*. Pokud věk bude menší nebo roven 45, pak s pravděpodobností 95 procent u osoby nebude mozkové mrtvice. Současně tak byl získán terminální uzel neboli list. Jestliže podmínka je splněna, strom rekurzivně provede proceduru a vytvoří nové uzly. V dalším kroku vidíme, že pokud věk bude větší jak 68 (tj. podmínka je splněná), pak s 26 procentní pravděpodobností nastane mozková mrtvice. Pokud pozorování nesplňují podmínku, jdeme na další uzel, kde obsah dat už je 33 procenta a rozhodování proběhne stejným způsobem podle podmínky *heart_ disease = ANO*.

Strom byl zkonstruován, pozorování byla roztříděna do podoblastí dle podmínek na jednotlivé proměnné, co dál? Nyní přichází na řadu predikce. Chceme zjistit, jak dobře si strom se svým klasifikačním úkolem poradil a je-li třeba jej nějak dále upravit.

K realizaci tohoto plánu používáme funkci *predict ()*. Poté data zobrazíme v tabulce.

Uděláme to pomocí funkce table().

```
> table_mat
prognoza
     ANO      NE

ANO  32       6

NE   14       49
```

Je zřejmé, že model zařazuje do kategorií poměrně dobře, viz počty pozorování na diagonále výše uvedené tabulky. Přesnost klasifikace je tedy následující:

```
[1] "Presnost testu je 0.801980198019802"
```

Toto byl výsledek pro jednu konkrétní volbu trénovacích a testovacích dat. Pro 1000 simulovaných datových souborů takto dostaneme přesnost klasifikace

```
> avarage/1000
[1] 0.7276238
```

tedy přibližně 73 procent.

Tím chci říci, že ačkoliv je model pro konkrétní volbu trénovacích a testovacích dat poměrně přesný, neznamená, že s jiným výběrem dat (resp. jejich rozdělením na trénovací a testovací) dosáhneme nutně stejného výsledku (přesnost pro jinou volbu vyšla například na 63 procent). Vše závisí právě na konkrétní volbě testovacích a trénovacích dat. Pro věrohodnější představu o přesnosti modelu je proto potřeba tento výběr opakovat, jak jsme to provedli výše. Elegantnější způsob pomocí tzv. křížové validace si potom představíme v následující kapitole.

Kapitola 3

Regresní stromy

Čerpala jsem informace pro tuto kapitolu z takových zdrojů, jako: [9], [5].

Jak bylo uvedeno v první kapitole, regresní rozhodovací stromy pracují se kvantitativními (závisle) proměnnými. Nyní bych se chtěla blíže podívat na základní principy a odlišnosti od klasifikačních stromů.

Pokud mluvíme o konstrukci regresních stromů, tak nakonec dostaneme přibližně stejné schéma. Proto v tomto problému přejdu přímo k implementaci algoritmu CART.

3.1. CART v regresních stromech

3.1.1. Algoritmus rozdělení dat

Stejně jako u klasifikačních stromů je hlavním úkolem modelu zkonstruovat strom, který by byl zcela dobře interpretovatelný a měl současně dobrou predikční schopnost. Na rozdíl od klasifikace, kde jsme za účelem hodnocení modelu využili Giniho koeficient a entropii (2.1, 2.2), zde bude hrát důležitou roli reziduální součet čtverců RSS (Residual Sum of Squares). Je tedy jasné, že za úkol máme minimalizovat tuto hodnotu,

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{3.1}$$

kde:

y_i - i-tá pozorovaná hodnota.

$\hat{y}_{R_j}$ - průměrná hodnota závisle proměnné trénovacích pozorování z j-té podmnožiny.

Postup probíhá takto: předpokládejme, že máme určitý prostor vysvětlujících proměnných, které později chceme rozdělit do některých podoblastí (obdélníkového tvaru), aby později mohl být tento model snadno interpretován. Nyní je tedy naším cílem najít podoblasti, které by minimalizovaly chybu RSS. Na každou z podoblastí aplikujeme vztah $\sum_{i \in R_j}(y_i - \hat{y}_{R_j})^2$. Potom aplikujeme finální „vnější“ součet, který sčítá všechny hodnoty, které jsme dostali pro jednotlivé podoblasti. Jedinou nevýhodou tohoto postupu je, že je velmi obtížné hledat obecně takovéto optimální podoblasti. A proto v tuto chvíli model používá tzv. „chamtivý“ algoritmus (z angl. greedy algorithm). Jak to funguje?

Chamtivé algoritmy jsou algoritmy, které předpokládají, že lokálně optimální řešení v každém kroku vedou k optimálnímu celkovému řešení. V tomto případě to znamená, že pokud byl atribut (proměnná) jednou vybrán a rozdělen na podoblasti, algoritmus se nemůže vrátit zpátky a vybrat jinou možnou proměnnou, která by poskytla nejlepší výsledné dělení. Proto ve fázi konstrukce nelze říci, zda vybraná proměnná nakonec poskytne optimální dělení.

3.1.2. Prořezávání

V předchozí kapitole jsem tuto metodu neuváděla pouze z důvodu, že data, se kterými jsem pracovala, nebyla tak obsáhlá (nevedla ke komplikaci rozhodovacího stromu). V regresním problému bych ráda uvedla příklad právě takového stromu, na který bych mohla tuto metodu aplikovat.

O se tedy jedná? Například jsme sestavili strom, který se zdá být v pořádku, ale jakmile vytvoříme jeho graf, zjistíme, že je obtížně interpretovatelný. Je příliš složitý a nesrozumitelný, s mnoha větvemi a uzly. Co v takové situaci dělat? Je nutné se obrátit na takovou metodu, jako je prořezávání. Skládá se z následujícího:

Nechť $|T|$ je počet koncových listů stromu T. Definujme $C\alpha(T)$ jako:

$$C\alpha(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha * |T|. \tag{3.2}$$

Každé hodnotě parametru α odpovídá takový prořezaný strom $T \subset T_0$, pro který hodnota $C\alpha(T)$ je minimální. Parametr α je tady představen v roli kontroly rovnováhy stromu mezi složitostí a přesností. Kompletní hodnocení stromu se tedy skládá ze dvou složek - chyby stromu (ve smyslu RSS) a penalizace za jeho složitost. Ve výsledku tak může být preferovaný méně rozvětvený strom, který dává větší chyby, před stromem, který dává menší chyby, ale je více rozvětvený.

Čím vyšší je hodnota parametru α, tím více penalizovaný strom s více listy. Obvykle u menších stromů dosáhne hodnota $C\alpha(T)$ minima. Pro optimální výběr hodnot parametru α se obvykle používá křížová validace, která bude představena v příští kapitole. Potom se vrátíme k původní datové sadě a na konec získáme oříznutý strom založený na volbě parametru α.

Příklad na prořezávání

Algoritmus CART se dívá na prořezávání, jako na hledání kompromisu mezi dvěma problémy: dosažení stromu optimální velikosti a také na získání co nejpřesnější klasifikace pozorování. Problém, který se pak může nastat, je velké množství podstromů, vzniklých po prořezávání jednoho stromu. Z tohoto důvodu chceme zkoumat pouze „nejlepší představitele“ pomocí následujícího oceňování:

Nechť $|T|$ je počet koncových listů stromu T, a pro ztrátovou funkci stromu $C\alpha(T)$ platí:

$$C_\alpha(T) = \underbrace{R(T)}_{\text{chyba klasifikace}} + \underbrace{\alpha * |T|}_{\text{penalizace za složitost}}, \qquad \alpha = [0, +\infty) \tag{3.3}$$

Poznámka 3.1 Je-li chyba klasifikace stromu se nemění, pak se během růstu stromu funkce $C_\alpha(T)$ bude zvyšovat. Z toho plyne, že je-li strom méně rozvětvený (ale dává velkou chybu klasifikace), může stát méně, něž strom, který dává menší chybu klasifikace, ale je víc rozvětvený.

Definujme T_{max} – maximální velikost neprořezávaného stromu. Pokud zavedeme fixní hodnotu α, pak existuje nejmenší minimalizovaný podstrom T_α, který splňuje následující podmínky:

$$C_\alpha(T(\alpha)) = min_{T \leq T_{max}} C_\alpha(T). \tag{3.4}$$

Tato podmínka nám říká, že neexistuje takový podstrom stromu T_{max}, který by měl pro tuto hodnotu α nižší cenu než $T(\alpha)$.

$$\text{if } C_\alpha(T) = C_\alpha(T(\alpha)) \text{ pak } T(\alpha) \leq T. \tag{3.5}$$

Druhá podmínka říká, že pokud existuje více než jeden podstrom, které mají danou $C_\alpha(T)$, pak vybereme nejmenší strom.

Přestože má nekonečný počet hodnot, existuje konečný počet podstromů stromu T_{max}. Lze sestavit sekvenci klesajících podstromů stromu T_{max}:

$$T_1 > T_2 > T_3 > ... > \{t_1\}, \tag{3.6}$$

(kde $\{t_1\}$ je kořenový uzel stromu) tak, že T_k je nejmenší minimalizovaný podstrom pro $[\alpha_k, \alpha_{k+1})$.

Další strom v pořadí můžeme získat, budeme-li používat prořezávání k aktuálnímu stromu v posloupnosti, tj. prořezáváním T_1 dostaneme strom T_2, atd.. S touto myšlenkou, půjdeme na algoritmus pro nalezení nejmenšího minimalizovaného podstromu pro různé hodnoty α.

První strom v této posloupnosti je nejmenší podstrom stromu T_{max}, který má stejnou chybu jako T_{max}, tj. $T_1 = T(\alpha = 0)$, jelikož dělení pokračuje, dokud v každém uzlu nezůstane pouze jedna třída, pak $T_1 = T_{max}$.

Naším úkolem tedy je nalézt T_1 z T_{max}. Postupujeme tak, že nalezneme libovolný pár listů se společným předkem, který by mohli sjednotit, tj. mohli bychom prořezat v rodičovský uzel bez zvýšení klasifikační chyby. Pokračujeme do toho momentu, pokud takovéto páry listů neeliminujeme. Takovým způsobem dostaneme strom, který má méně větví, než T_{max}, ale hodnota jeho ztrátové funkce se nezměnila.

Další otázka, se kterou se setkáme, zní následovně: jak nalézt další stromy v posloupnosti - $T_2, T_3, T_4, ...$ a odpovídající hodnoty α?

Označme T_t - větev stromu T s kořenovým uzlem t. Musíme najít odpověď na otázku: při kterých hodnotách bude strom $T - T_t$ lepší než T. Budeme-li prořezávat v uzlu t, pak jeho příspěvek v plnou cenu stromu $T - T_t$ bude:

$$C_\alpha(\{t\}) = R(t) + \alpha, \tag{3.7}$$

kde $R(t) = \frac{m}{n}$, m - počet špatně klasifikovaných pozorování a n - celkové množství klasifikovaných pozorování pro celý strom.

Příspěvek T_t k celkové ceně stromu T bude

$$C_\alpha(T_t) = R(T_t) + \alpha|T_t|, \tag{3.8}$$

kde $R(T_t) = \sum_{t' \in T_t} R(t')$ - suma chyb klasifikace s kořenovým uzlem t.

Strom $T - T_t$ bude lepší než T, když $C_\alpha(\{t\}) = C_\alpha(T_t)$, protože při této hodnotě mají stejnou cenu, ale $T - T_t$ je menší z těchto dvou.

Když $C_\alpha(\{t\}) = C_\alpha(T_t)$ pak:

$$R(T_t) + \alpha|T_t| = R(t) + \alpha. \tag{3.9}$$

Řešením pro α dostaneme:

$$\alpha = \frac{R(t) - R(T_t)}{|T_t| - 1}. \tag{3.10}$$

Předchozí úvahy je možné shrnout do následujícího algoritmu:

Postup:

1) Inicializace:

a) nechť T_1 je strom, který jsme dostali při $\alpha_1 = 0$.

2) Krok první:

a) vybrat uzel $t \in T_1$, který minimalizuje $g_1(t) = \frac{R(t) - R(T_1, t)}{|T_1, t| - 1}$,

b) nechť t_1 je tento uzel,

c) nechť $\alpha_2 = g_1(t_1)$ a $T_2 = T_1 - (T_1, t)$.

3) Krok i-tý:

a) vybrat uzel $t \in T_i$, který minimalizuje $g_i(t) = \frac{R(t) - R(T_i,t)}{|T_i,t|-1}$,

b) nechť t_i je tento uzel,

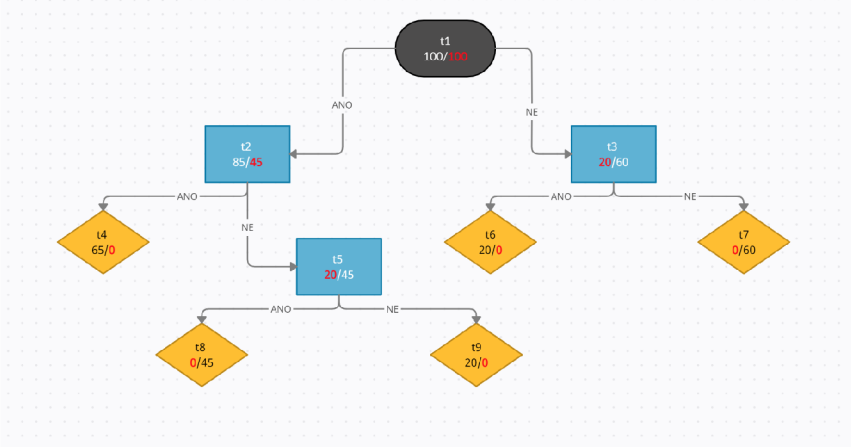
c) nechť $\alpha_{i+1} = g_i(t_i)$ a $T_{i+1} = T_i - (T_i, t)$.

Na výstupu budeme mít posloupnost stromů a odpovídající posloupnost parametrů α.

$$T_1 \supseteq T_2 \supseteq ... \supseteq T_k \supseteq ... \supseteq \{t_1\} \text{ a}$$
$$\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_k \leq ....$$

Parametr $\alpha \in [\alpha_k, \alpha_{k+1})$ přitom vybíráme například metodou křížové validace.

Nechť nám je dán rozhodovací strom (obrázek 3.1):



Obrázek 3.1: *Počáteční strom, který nevede k žádnému chybně klasifikovanému pozorování.*

Chceme spočítat $g_1(t)$ pro všechny uzly t v T_1:

$$g_1(t_1) = \frac{R(t_1) - R(T_1,t_1)}{|T_1,t_1|-1}.$$

Celkem byl strom natrénován pro 200 pozorování, pocházející ze dvou skupin po 100 pozorování (100 + 100 = 200 pro kořenový uzel), tedy $R(t_1) = \frac{m}{n}$ pro $m = 100, n = 200$, odtud $R(t_1) = \frac{1}{2}$.

$R(T_1, t_1)$ - podíl chybně klasifikovaných pozorování u všech listů podstromu. Počítá se to tak, že vezmeme součet chybně zařazených pozorování v listech v poměru k celkovému počtu pozorování pro strom. V příkladu vše dělíme 200. Protože pro podstrom s kořenem v t_1 je to také strom T_1, všechny listy mají správně klasifikovaná pozorování, proto

$$R(T_1, t_1) = \sum_{1}^{5} \frac{0}{200} = 0.$$

Tady můžeme taky říci, že původní strom T_{max} se nezměnil od stromu T_1 protože neměl v listech špatně klasifikované pozorování. Dále, nechť $|T_1, t_1|$ je počet listů podstromu s kořenem v uzlu t_1. Dohromady takovýchto máme 5.

Dostaneme:

$g_1(t_1) = \frac{(\frac{1}{2} - 0)}{(5-1)} = \frac{1}{8}$,

$g_1(t_2) = \frac{(\frac{45}{200} - 0)}{(3-1)} = \frac{9}{80}$,

$g_1(t_3) = \frac{(\frac{20}{200} - 0)}{(2-1)} = \frac{1}{10}$,

$g_1(t_5) = \frac{(\frac{20}{200} - 0)}{(2-1)} = \frac{1}{20}$.

Uzel t_5 má minimální hodnotu g_1, a tedy dostaneme nový strom T_2, pomocí prořezávání uzlu t_5. To jest, naše α je nejmenší hodnota g_1 pro dané uzly. V našem případě to je pro uzel t_5.

Pokračujeme ve výpočtech hodnot g pro T_3:

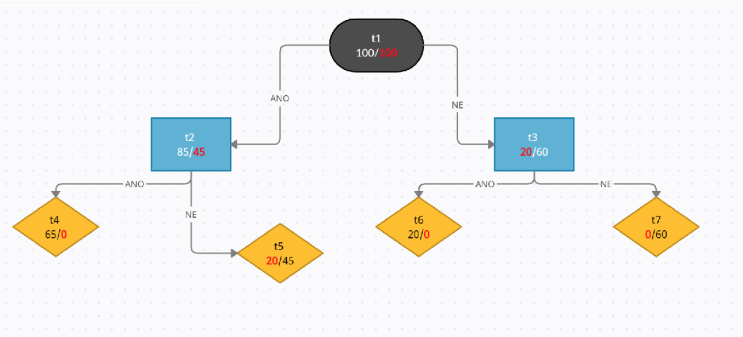
$g_2(t_1) = \frac{(\frac{1}{2} - (\frac{0}{200} + \frac{0}{200} + \frac{0}{200} + \frac{20}{200}))}{(4-1)} = \frac{2}{15}$,

$g_2(t_2) = \frac{(\frac{45}{200} - (\frac{20}{200} + \frac{0}{200}))}{(2-1)} = \frac{1}{8}$,

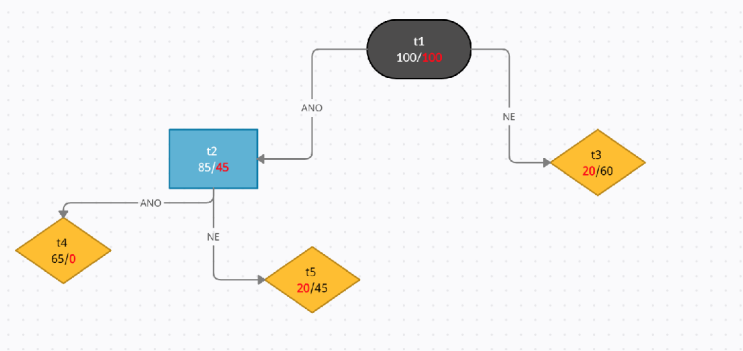
$g_2(t_3) = \frac{(\frac{20}{200} - (\frac{0}{200} + \frac{0}{200}))}{(2-1)} = \frac{1}{10}$.

Tentokrát má uzel t_3 minimální hodnotu g_2, a dostaneme nový strom T_3, pomocí prořezávání uzlu t_3.

Pokračujeme ve výpočtech hodnot g pro T_4,



Obrázek 3.2: *Strom, který jsme dostali po prořezávání uzlu t_5. Tady vidíme, že teď už v* ***listu*** *t_5 máme 20 chybně klasifikovaných pozorování.*



Obrázek 3.3: *Tady prořezáváme uzel t_3 a vidíme, že v listu t_3 máme ještě 20 chybně klasifikovaných pozorování.*

$$g_3(t_1) = \frac{(\frac{1}{2} - (\frac{0}{200} + \frac{20}{200} + \frac{20}{200}))}{(3-1)} = \frac{3}{20},$$

$$g_3(t_2) = \frac{(\frac{45}{200} - (\frac{0}{200} + \frac{20}{200}))}{(2-1)} = \frac{1}{8},$$

A tedy vidíme, že minimum je v t_2.

Poslední výpočet máme pro strom T_5:

$$g_4(t_1) = \frac{(\frac{1}{2} - (\frac{45}{200} + \frac{20}{200}))}{(2-1)} = \frac{7}{40}.$$

Minimum máme v uzlu t_1, z tohoto důvodu ho prořezáme a dostaneme kořen stromu ($T_5 = \{t_1\}$). Na tomto se proces prořezávání zastaví. Dostaneme posloupnost: $\alpha_1 = 0$, $\alpha_2 = \frac{1}{20}$, $\alpha_3 = \frac{1}{10}$, $\alpha_4 = \frac{1}{8}$, $\alpha_5 = \frac{7}{40}$. Na každém kroku jsme hledali α, které by bylo nejlepší pro každý strom v klesající posloupnosti (3.6). Role α v dalším textu je snadno pochopitelná (vyjadřuje optimální hodnotu mezi složitostí a chyb) (3.5).

V důsledku máme, že T_1 je nejlepší strom pro $[0, \frac{1}{20})$, T_2 pro $[\frac{1}{20}, \frac{1}{10})$, T_3 pro $[\frac{1}{10}, \frac{1}{8})$, T_4 pro $[\frac{1}{8}, \frac{7}{40})$ a T_5 pro $[\frac{7}{40}, \infty)$.

3.1.3. Křížová validace

V kapitole 2.2 bylo ukázáno, jak se vyhnout zkreslenému vyhodnocení přesnosti klasifikace pomocí funkce *sample()*. V této části bych chtěla mluvit o elegantnějším odhadu přesnosti klasifikace pomocí křížové validaci. Nejprve pojďme zjistit, co to vlastně je.

Křížová validace je metoda pro vyhodnocení analytického modelu a jeho chování na testovacích datech s využitím dostupných dat co nejrovnoměrněji.

Jak proces probíhá? Pro začátek rozdělíme původní data na přibližně stejné bloky. Vezměme například $k = 7$. Potom postupně snížíme počet těchto bloků o $k-1$. Ukázalo se, že model je trénován na šesti blocích a sedmý blok se používá k testování. Postup se opakuje k-krát a při každém průchodu se vybere nový blok k ověření a na zbývajících se provede natrénování modelu. Křížové ověření má důležitou výhodu oproti použití jednoho datového souboru pro trénování a jednoho pro testování modelu: pokud je pro každý průchod odhadnuta výstupní chyba modelu (tedy chyba na testovacích datech z jednoho z bloků) a zprůměrována ve všech průchodech, pak bude výsledný odhad spolehlivější a méně zatížený variabilitou plynoucí z náhodného výběru trénovacích, resp. testovacích dat. Typicky se následně výběr bloků opakuje, aby bylo dosaženo co nejstabilnějších výsledků.

3.2. Příklad na regresní stromy

Pro ilustraci konstrukce a použití regresních stromů jsem využila balíček Wine Red, který popisuje kvalitu různých druhů červených vin [2]. Mým cílem je určit, jak sklad vína ovlivňuje její kvalitu. Máme také proměnné, jak: alkohol, sírany, těkavá kyselost, pevná kyselost, pH, celkový oxid siřičitý, volný oxid siřičitý, hustota a kyselina citronová. Dohromady máme 998 pozorování.

Na začátku jsem připojila příslušné knihovny statistického softwaru R, data z excelovského souboru a rozdělila data náhodně na dvě poloviny, trénovací a testovací. Tato proměnná *train* obsahuje pouze trénovací data.

Konstruujeme strom, kde je cílovou funkcí (závisle proměnnou) hodnocení vína. Úkol je následující: jaké parametry obsahu vína ovlivňují jeho vysokou nebo nízkou kvalitu? Pomocí funkcí *plot()* a *test()* nejprve postavíme kostru stromu, načež přidáme samotný text.

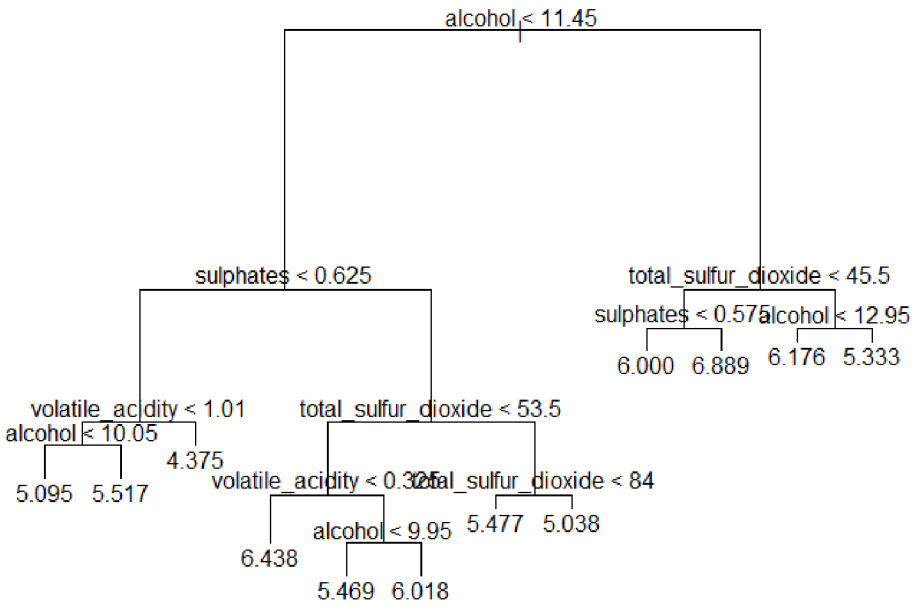
```
Regression tree:
tree(formula = quality ~ ., data = winequality_red, subset = train)
Variables actually used in tree construction:
[1] "alcohol"          "sulphates"        "volatile_acidity"
"total_sulfur_dioxide"
Number of terminal nodes:  12
Residual mean deviance:  0.2963 = 144.3 / 487
Distribution of residuals:
Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.51700 -0.43750 -0.09494  0.00000  0.48280  1.98200
```

Vidíme, že strom obsahuje 4 proměnné („alcohol“, „sulphates“, „volatile acidity“, „total sulfur dioxide“) a 12 listů. Hodnota reziduálního součtu čtverců (RSS) je 0.2963. Nyní zkonstruujeme samotný regresní strom a ohodnotíme jeho složitost a interpretovatelnost.

Už po prvním pohledu na tento graf lze říci, že rozhodující pro kvalitu vína bude obsah alkoholu. Důležitou roli hrají i další proměnné, každopádně strom je poměrně složitý a tedy jako celek i hůře interpretovatelný. Zvážíme tedy jeho prořezání. V takovém případě budeme používat proceduru, která byla popsaná výše, viz strana 29.

Teď se zaměříme na zlepšení samotného stromu. Když se podíváme na obrázek 3.4, můžeme si všimnout, že došlo k přeškolení modelu, ve smyslu stromu obvykle nám o tom říká přešil velká výška stromu, a tedy musíme ostřihnout větví. První otázka, která se vznikne ve hlavě je: kolik vlastně máme ostřihnout větev? Tady musíme použit křížovou validaci. Vyzkoušíme ji.

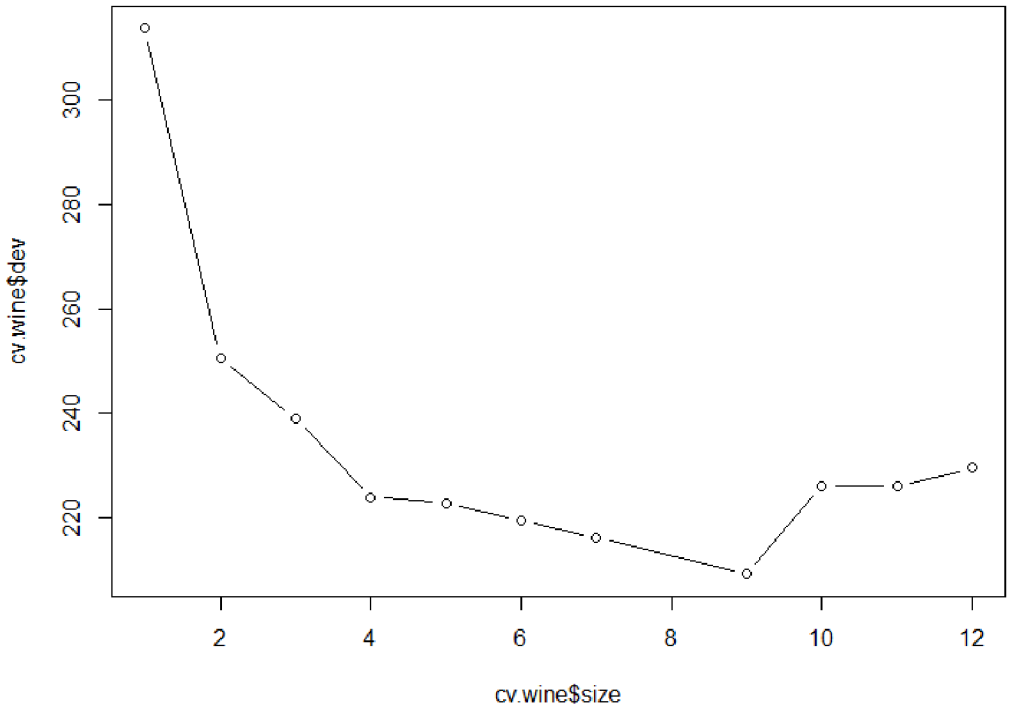
Standardně se používá kolem 10 k-násobení, ale klidně může být jich více,



Obrázek 3.4: *Výsledný regresní strom pro data týkající se kvality červených vín.*

nebo méně. Podíváme se na obrázek 3.5. Na ose x máme hodnoty, které popisují rozměrnost stromu, a na ose y - přípustné chyby. Naším cílem je tedy najít na grafu hodnotu, pří které chyba bude minimální. Pouhým okem vidíme hodnotu 9, která vlastně je naším α.

Sestavíme tento strom (obrázek 3.6). Můžeme teď říct něco o výsledku. Jak vidíme na obrázku, hlavní proměnnou je obsah alkoholu v červeném víně, a pokud je jeho hodnota striktně menší než 11.45, pak přejdeme do uzlu, ve kterém je rozhodovací pravidlo založeno na síranech (sulphates), a pokud je již méně než 0.625, pak jdeme na těkavé kyselosti (volatile acidity), kde již vidíme výrazný rozdíl v kvalitě (skoro jeden bod). Dál vidíme zase rozhodovaní na základě alkoholu, a tam se to už výrazně neliší. Pokud nás zajímá nejlepší kvalita, pak se můžeme podívat na pravou stranu stromu. Tam vidíme číslo 6.889. Dostaneme



Obrázek 3.5: *Zobrazuje celé chování stromu a taky hodnotu, kterou budeme používat pro prořezávání.*

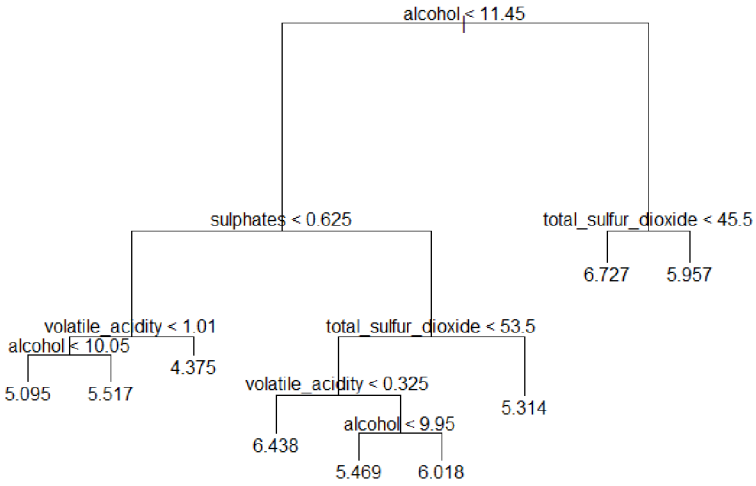
tuto kvalitu, pokud v vině bude alkoholu méně, jak 11.45 a zároveň celkového oxidu siřičitý (total sulfur dioxide) bude méně, jak 45.5 a síranů více, jak 0.57.

Na konci by to ještě chtělo analýzu kvality predikce. Budu postupovat následovně: přidám zvlášť do kódu funkce *while*, která spočítá průměrnou přesnost našeho modelu pro 1000 náhodně vygenerovaných stromů.

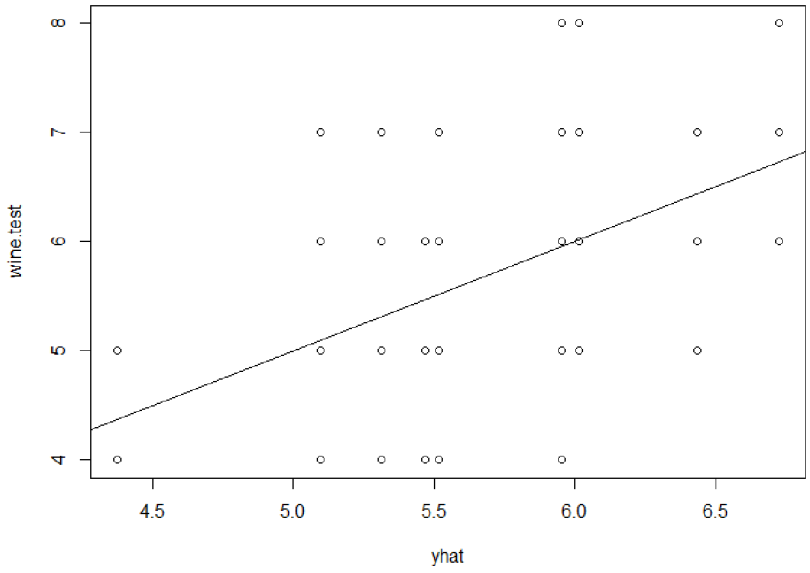
```
> avarage/1000
[1] 0.6941557
```

Podle výsledku můžeme říct, že chyba je velká, skoro až jeden bod, to způsobeno tím, že samotné data mají velkou variabilitu.

Chybu jsme určili podle průměru neprořezávaných stromů. Pokusíme se zlepšit výsledek na základě prořezávaného stromu. Porovnám dvě hodnoty.



Obrázek 3.6: *Finální strom po prořezávání.*



Obrázek 3.7: *Srovnání pozorovaných a predikovaných hodnot závislé proměnné.*

```
[1] 0.6940891
[1] 0.6168614
```

Vidíme, že ostřihnuti pomohlo jen trochu. Odchylku od kvality stálé máme dost vysokou.

Kapitola 4

Bagging a náhodný les

4.1. Bagging

V teto kapitole jsem použila tuto literaturu: [3].

Pokud provedeme hodnocení při vytváření rozhodovacích stromů s různými výběry dat pro trénování a testování, můžeme si všimnout, že všechny mají vysokou variabilitu. To znamená, že každý strom je někdy velmi odlišný od druhého. Tento problém řeší metoda bagging, stejně jako metoda náhodného lesa.

Jak funguje bagging? Pro daná trénovací data provedeme bootstrap. To znamená, že z pozorování obsažených v trénovacích datech provedeme nový výběr s vracením o stejném rozsahu. Následně trénujeme naši metodu na tomto výběru, abychom našli predikci $\hat{f}^{*b}(x)$ a pak zprůměrujeme všechny naše předpovědi a dostaneme:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x), \tag{4.1}$$

Přitom vycházíme z úvahy, že se tímto způsobem, analogicky jako třeba u výběrového průměru, sníží variabilita predikce.

Bagging lze použít jak v regresních problémech, tak i v klasifikačních. Pokud chceme aplikovat regresní metody, stačí pouze vytvořit B regresních stromů, které jsou založené na B trénovacích výběrech a pak provést stejný postup, tj. zprůměrovat hodnoty predikce. Pokud jde o klasifikační stromy, postup je odlišný.

Pro dané testovací pozorování zaznamenáme třídu, kterou předpovídá každý ze stromů B, a pak aplikujeme řešení, které je založeno na „volbě většiny“. To znamená, že nakonec je rozhodnutí (zařazení do výsledné třídy) založeno na tom, která předpověď bude nejčastěji.

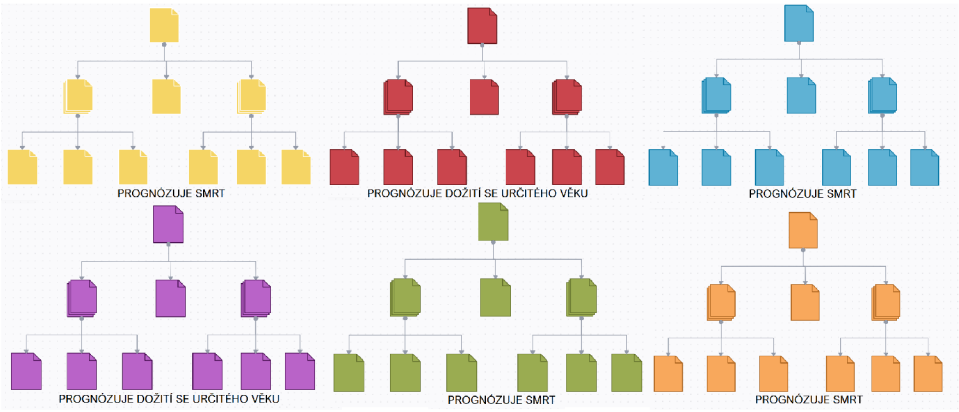
Odhad chyby ze zbývajících dat. Je známo, že při konstrukci stromů metodou bagging se používají pouze dvě třetiny všech trénovacích pozorování (odpovídá to pravděpodobnosti zahrnutí daného pozorování v bootstrapovém výběru, která je 0.632). Taková pozorování se nazývají zbývající pozorování. Vezmeme si pozorování, která nám zbyla ze stromů, které jsme už zahrnuli do výběru, a pro ně predikujeme hodnotu závisle proměnné. V důsledku toho získáme předpovědi přibližně $B/3$-krát pro každé trénovací pozorování. Pak již následuje známý postup, který je rozdělen do dvou možných kroků. V případě regresních stromů se pro dané pozorování vezme průměrná predikovaná hodnota a v případě klasifikačních stromů se o přiřazení do třídy rozhodne na základě většinové volby. Nakonec potom spočítáme průměrnou chybu predikce ze všech (trénovacích) pozorování.

4.2. Náhodný les

Co to tedy náhodný les vlastně je a jak funguje? Technicky řečeno, náhodný les definujeme jako soubor jednotlivých rozhodovacích stromů. V jazyce datové vědy existuje důvod, proč model náhodného lesa funguje tak: velký počet relativně nekorelovaných stromů spolupracujících společně, překoná případné predikční slabosti jednotlivých stromů.

Na obrázku číslo 4.1 můžeme pozorovat následující situaci: máme dohromady šest rozhodovacích stromů, které reprezentují náhodný les. Každý ze stromů má pro dané pozorování svou předpověď. Žlutý, modrý, zelený a oranžový prognózují smrt nějakého jedince. Mezi tím červený a fialový prognózují dožití se určitého věku. Jelikož máme čtyři hlasy za to, že jedinec umře a dva hlasy za to, že bude žít, zvítězí prognóza smrti.

Klíčovým bodem je nízká korelace mezi stromy. Proč je to dobré? Nesouvise-



Obrázek 4.1: *Reprezentací náhodného lesu. Máme šest různých stromů, které prognózují na základě svých trénovacích sad.*

jící modely mohou vytvářet souborné předpovědi, které jsou přesnější než jakýkoli jednotlivý model. Důvodem tohoto efektu je, že stromy chrání před individuálními chybami ostatních. Pokud některé stromy nefungují správně, jiné mohou být správné a provádět rozumné předpovědi, nadto s výrazně redukovanou chybou oproti použití jednotlivých stromu i baggingu. V návaznosti na tento koncept se stromy mohou pohybovat správným směrem jako skupina.

Náhodný les nyní stojí před následující otázkou: jak zaručit, že chování konkrétního stromu nekoreluje s chováním jiného stromu? Zde se využívá toho, že se pro konstrukci každého stromu, resp. každého dělení v něm, využívá pouze určitá podmnožina vysvětlujících proměnných (vždy jiná, náhodně zvolená). Při použití všech proměnných bychom takto dostali předchozí metodu baggingu.

Samozřejmě, v případě baggingu i náhodného lesa je potřeba počítat s tím, že jsou tyto metody výpočetně náročnější.

4.3. Příklad na bagging a náhodný les

Podívejme se na použití náhodného lesu a baggingu. Zde jsem zvolila datový soubor, který se zabývá problematikou pojištění obsahující 7 proměnných a 587 pozorování [4]. Tedy bych potřebovala určit chybu predikované ceny, která závisí na takových proměnných: kuřák, věk, pohlaví, dětí, bmi (index tělesné

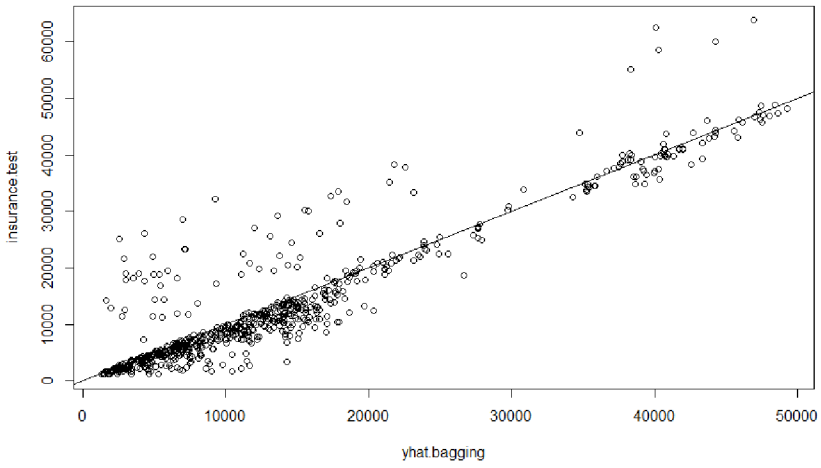
hmotnosti) a region. K tomu bych pak chtěla zjistit, jaka proměnná má největší vliv na konstrukce ceny jednotlivého pojištění.

```
Call:
Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 6

Mean of squared residuals: 24830372
% Var explained: 82.5
```

Nejprve připojíme knihovnu, kterou potřebujeme. Nyní jsme vytvořili testovací a trénovací sady s pozorováními. Zde $mtry = 6$ nám říká, že při vytvoření každého uzlu ve stromu, musíme vzít v úvahu 6 prediktorů, mimo "charges", jelikož jí používáme jako cílovou funkci podle které se nebude probíhat analýza. Jinými slovy, používáme bagging. Vytvořili jsme 500 stromů.

```
sqrt(chyba)
[1] 4688.709
```



Obrázek 4.2: *Závislost predikovaných hodnot a testovaných. Vidíme, že model bude predikovat praktický správně, protože variabilita není příliš vysoká.*

Zde chceme vyhodnotit náš model a zjistit, zda nám bagging pomůže mi-

nimalizovat chybu. Vidíme, že chyba je 4688.709. Na obrázku 4.2 pak můžeme vidět, že se porovnání nepozorovaných a predikovaných hodnot ani nevyznačuje příliš velkou variabilitou (což je samozřejmě dobře). Pro zajímavost: chyba, která vznikla při konstrukci jediného stromu, je rovna 13117.27.

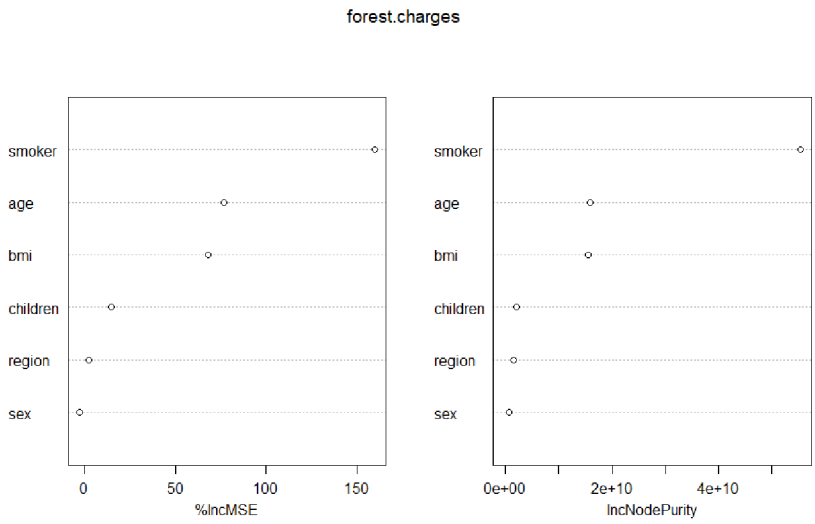
V dalším kroku jsme změnili počet stromů, které funkce standardně nastavuje. Můžeme vidět, jak chyba klesá.

```
sqrt(chyba)
[1] 4751.729
```

Standardně náhodný les využívá menší hodnotu argumentu ***mtry***, a vlastně: pro regresní stromy platí $mtry = \frac{p}{3}$, a pro klasifikační - $mtry = \sqrt{p}$. Nastavíme $mtry = 3$. To pomohlo chybu ještě dále zredukovat. Takže konečná chyba po všech manipulacích je rovna 4612.759. Můžeme říct, že náhodný les vede k menší chybě, čím bagging.

Poslední dva grafy na obrázku 4.3 nám ukáží, které proměnné zde hrají důležitou roli. I bez vytvoření grafu by se hodnocením předchozích stromů dalo říci, že nejdůležitějším parametrem je zde proměnná „smoker“, jelikož cigarety vážně ovlivňují zdraví. Na prvním grafu vidíme počet prediktorů sestupně od nejdůležitějšího, a dolů - procent střední kvadratické chyby. Na druhém grafu vidíme dolů veličinu, která popisuje nečistotu uzlů, tj. jak dobře stromy rozdělují data. Používáme Giniho index na určování důležitosti v proměnné (střední pokles nečistot uzlu), anebo chybu klasifikace (střední pokles přesnosti).

Nyní bych chtěla říci, že jak už bylo uvedeno na začátku, že rozhodovací stromy obecně nepředstavují nejefektivnější nástroj pro predikci, resp. klasifikaci. Možná vylepšení základního modelu, zde reprezentovaná baggingem a náhodnými lesy, jsou zase výpočetně poměrně náročná. Velkou předností rozhodovacích stromů je ovšem jejich snadná interpretovatelnost, což z nich činí v praxi oblíbený nástroj datové analýzy.



Obrázek 4.3: *Grafy, zobrazující důležitost proměnných při analýze.*

Závěr

V závěru své práce o rozhodovacích stromech bych ráda podotkla, že základní myšlenka rozhodovacích stromů je opravdu poměrně intuitivní, což je v praxi velmi výhodné při prezentaci a interpretaci výsledků. Jakmile jsem do tohoto tématu ponořila, naučila jsem se spoustu nových a zajímavých věcí.

Dá se dokonce říci, že každý člověk podvědomě používá algoritmus rozhodovacích stromů ve své hlavě – každý den činí rozhodnutí, která jsou založena na některých událostech v jeho životě, a to je pak takovou stromovou strukturou dobře reflektováno. Ačkoli tento model ne vždy vede k absolutně přesným předpovědím, kompenzuje to interpretací a možností aplikace v jakémkoli aspektu vědy a v každodenním životě.

Literatura

[1] BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., STONE, C. J., (1984). Classification and regression trees. Monterey, CA: Wadsworth and Brooks/Cole Advanced Books and Software. ISBN 978-0-412-04841-8.

[2] CERDEIRA, A., ALMEIDA, F., MATOS, T. and REIS, J., Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal [cit. 17.06.2022]. Dostupné z: https://archive.ics.uci.edu/ml/datasets/wine+Quality

[3] JAMES, G., WITTEN, D., HASTIE, T., TIBSHIRANI, R. An Introduction to Statistical Learning with Applications in R. Springer, New York, 2013.

[4] The Open Knowledge Foundation., 2017. Medical Cost Personal Datasets[online] [cit. 17.05.2022]. Dostupné z: https://www.kaggle.com/datasets/mirichoi0218/insurance

[5] Penn State Mark, 2022. Minimal Cost-Complexity Pruning. In: Psu.edu/[online].[cit. 17.06.2022]. Dostupné z: https://online.stat.psu.edu/stat508/lesson/11/11.8/11.8.2

[6] Rozhodovací stromy v R. In: Coderlessons.com[online].[cit. 21.11.2021]. Dostupné z: https://coderlessons.com/tutorials/mashinnoe-obuchenie/r-programmirovanie/28-derevia-reshenii-v-r

[7] Stroke Prediction Dataset[online] [cit. 17.08.2022]. Dostupné z: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

[8] SHAHIDI, A., 2019. Rozhodovací stromy: Obecné principy. In: Loginom.ru[online].[cit. 9.11.2021]. Dostupné z:

https://loginom.ru/blog/decision-tree-p1

[9] VARMUZA, K., FILZMOSER, P., 2009. Introduction to Multivariate Statistical Analysis in Chemometrics. 2 vyd. USA Taylor and Francis Group. ISBN 978-1-4200-5947-2.

Příloha A

Kódy v R k jednotlivým tématům

Kód ke klasifikačnímu stromu

```
set.seed(7)
library(tree)
library(rpart)
library(rpart.plot)
library(readr)
BrainStrokeData <- read_csv("C:/photoAndData/brainstroke1.csv", col_types
BrainStrokeData = na.omit(BrainStrokeData)
BrainStrokeData$stroke = ifelse(BrainStrokeData$stroke == 0, "NE", "ANO"
BrainStrokeData$heart_disease = ifelse(BrainStrokeData$heart_disease ==
BrainStrokeData$hypertension = ifelse(BrainStrokeData$hypertension == 0,
BrainStrokeData$id = NULL
View(BrainStrokeData)
head(BrainStrokeData)
tail(BrainStrokeData)
new_data = sample(1:nrow(BrainStrokeData))
BrainStrokeData = BrainStrokeData[new_data, ]
head(BrainStrokeData)
tail(BrainStrokeData)

head(BrainStrokeData)
trenovaci_testovaci = function(data, size = 0.8, train = TRUE) {
n_radek = nrow(data)
t_radek = size * n_radek
trenovaci_sestavene = 1:t_radek
if (train == TRUE) {
return (data[trenovaci_sestavene, ])
} else {
```

```
return (data[-trenovaci_sestavene, ])
}
}
trenovaci = trenovaci_testovaci(BrainStrokeData, 0.8, train = TRUE)
testovaci = trenovaci_testovaci(BrainStrokeData, 0.8, train = FALSE)
dim(trenovaci)
dim(testovaci)
prop.table(table(trenovaci$stroke))
strom = rpart(trenovaci$stroke~., data = trenovaci, method = 'class')
rpart.plot(strom)

prognoza = predict(strom, testovaci, type = "class")
table_mat = table(testovaci$stroke, prognoza)
table_mat

presnost = sum(diag(table_mat)) / sum(table_mat)
print(paste('Presnost testu je', presnost))



n = 0
avarage = 0
while(n<1000) {
new_data = sample(1:nrow(BrainStrokeData))
BrainStrokeData = BrainStrokeData[new_data, ]
trenovaci_testovaci = function(data, size = 0.8, train = TRUE) {
n_radek = nrow(data)
t_radek = size * n_radek
trenovaci_sestavene = 1:t_radek
if (train == TRUE) {
        return (data[trenovaci_sestavene, ])
} else {
        return (data[-trenovaci_sestavene, ])
}
}
trenovaci = trenovaci_testovaci(BrainStrokeData, 0.8, train = TRUE)
testovaci = trenovaci_testovaci(BrainStrokeData, 0.8, train = FALSE)
strom = rpart(trenovaci$stroke~., data = trenovaci, method = 'class')
prognoza = predict(strom, testovaci, type = "class")
table_mat = table(testovaci$stroke, prognoza)
presnost = sum(diag(table_mat)) / sum(table_mat)
presnost
avarage = avarage + presnost
```

```
n=n+1
}
avarage/1000
```

Kód k regresnímu stromu

```
set.seed(1)
library(tree)
library(ISLR)
library(rpart)
library(rpart.plot)
library(readxl)
library(caret)
winequality_red <- read_csv("C:/photoAndData/winequality-red.csv")
winequality_red$alcohol = as.numeric(winequality_red$alcohol)
winequality_red  = na.omit(winequality_red)
train = sample(1:nrow(winequality_red), nrow(winequality_red)/2)
tree.wine = tree(quality~., winequality_red, subset = train)
summary(tree.wine)
plot(tree.wine)
text(tree.wine, pretty = 0)
cv.wine = cv.tree(tree.wine, K = 10)
plot(cv.wine$size, cv.wine$dev, type = "b")
plot(cv.wine)
prune.wine = prune.tree(tree.wine, best = 9)
plot(prune.wine)
text(prune.wine, pretty = 0)

n = 0
avarage = 0
while(n<1000) {
train = sample(1:nrow(winequality_red), nrow(winequality_red)/2)
tree.wine = tree(quality~., winequality_red, subset = train)
summary(tree.wine)
yhat = predict(tree.wine, newdata = winequality_red[-train, ])
wine.test = winequality_red[-train, 'quality']
wine.test = as.numeric(unlist(wine.test))
chyba = mean((yhat - wine.test)^2)
avarage = avarage + sqrt(chyba)
n=n+1
}
avarage/1000
```

```
%porovnani 2 hodnot%
yhat = predict(tree.wine, newdata = winequality_red[-train, ])
wine.test = winequality_red[-train, 'quality']
wine.test = as.numeric(unlist(wine.test))
plot(yhat, wine.test)
abline(0, 1)
MSE = mean((yhat - wine.test)^2)
odchylka_od_kvality = sqrt(MSE)
odchylka_od_kvality

yhat = predict(prune.wine, newdata = winequality_red[-train, ])
wine.test = winequality_red[-train, 'quality']
wine.test = as.numeric(unlist(wine.test))
plot(yhat, wine.test)
abline(0, 1)
MSE = mean((yhat - wine.test)^2)
odchylka_od_kvality = sqrt(MSE)
odchylka_od_kvality
```

Kód k baggingu a náhodnému lesu

```
library(readxl)
library(randomForest)
set.seed(9)
insurance = read.csv("C:/photoAndData/insurance.csv", sep = ",")

View(insurance)
insurance = na.omit(insurance)
train = sample(1:nrow(insurance), nrow(insurance)/2)
train = na.omit(train)
insurance.test = insurance[-train, 'charges']
set.seed(2)
bagging.charge = randomForest(charges~., data = insurance, subset = train
bagging.charge

yhat.bagging = predict(bagging.charge, newdata = insurance[-train, ])
length(insurance.test)
plot(yhat.bagging, insurance.test)
abline(0,1)
chyba = mean((yhat.bagging - insurance.test)^2)
sqrt(chyba)
```

```
bagging.charge = randomForest(charges~.,
data = insurance, subset = train, mtry = 6, ntree = 50)
bagging.charge
yhat.bagging = predict(bagging.charge, newdata = insurance[-train, ])
chyba = mean((yhat.bagging - insurance.test)^2)
sqrt(chyba)

forest.charges = randomForest(charges~.,
data = insurance, subset = train, mtry = 3, importance = TRUE)
yhat.forest = predict(forest.charges , newdata = insurance[-train, ])
chyba = mean((yhat.forest - insurance.test)^2)
sqrt(chyba)


importance(forest.charges )
varImpPlot(forest.charges )

summary(insurance)
```