VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

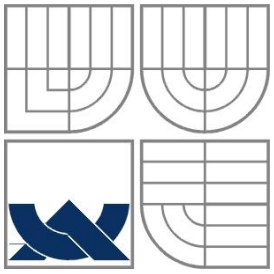
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

KLASIFIKAČNÉ METÓDY PRE DÁTA Z MIKROČIPOV

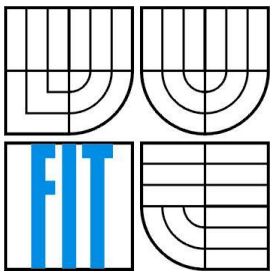
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. VLADIMíR HUDEC
AUTHOR

BRNO 2008
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

KLASIFIKAČNÍ METODY PRO DATA Z MIKROČIPŮ

CLASSIFICATION METHODS FOR MICRIARRAYS DATA

DIPLOMOVÁ PRÁCE

MASTER‘S THESIS

AUTOR PRÁCE — Bc. VLADIMíR HUDEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. IVANA BURGETOVÁ, Ph.D.
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce pojednává o datech získaných z genetických čipů a metodách jejich analýzy. Rozebírá některé metody pro analýzu těchto dat a detailněji se zaměřuje na metodu „Random Forests“. Obeznamuje s datasetem který je použit pro experimenty. Metody jsou realizovány v prostředí jazyka R. Jednotlivé výsledky jsou zhodnoceny a porovnány. Výsledky dosažené s metodou „Random Forests“ jsou porovnány s jinými experimenty nad stejným datasetem.

Abstract

This paper discusses about the data obtained from gene chips and methods of their analysis. Analyzes some methods for analyzing these data and focus on the method of "Random Forests". Shows dataset that is used for specific experiments. Methods are realized in R language environment. Than they are tested, and the results are presented and compared. Results with method "Random Forests" are compared with other experiments on same dataset.

Klíčová slova

microarray, Random Forests,

Keywords

microarray, Random Forests,

Citace

Hudec Vladimír: Klasifikačné metódy pre dáta z mikročipov, diplomová práca, Brno, FIT VUT v Brne, 2011

Klasifikačné metódy pre dáta z mikročipov

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Ivany Burgetové, Ph.D.
Uviedol som všetky literárne pramene a publikácie z ktorých som čerpal.

.....
Vladimír Hudec
25. máj 2011

Poďakovanie

Chcel by som sa poďakovať vedúcej práce Ing. Ivane Burgetovej, Ph.D. za odbornú pomoc a zabezpečenie školského výpočtového servera.

Obsah

Úvod

Bioinformatika je moderná vedná disciplína spájajúca biológiu a informatiku. Týka sa najmä využívania informačných technológií pre biologický výskum a vývoj. Predovšetkým pre analýzu obrovského množstva dát ktoré sa v oblasti biologického výskumu podarilo získať. Kombinácia týchto dvoch odborov už dopomohla veľkému množstvu objavov ktoré skvalitňujú život ľudí. Avšak ešte väčšie sú očakávania nových poznatkov, ktoré má bioinformatika potenciál poskytnúť. Jedným z odvetný kde je biológia s informatikov úzko prepojená je výskum ľudského genómu. Aj keď toto spojenie už existovalo skôr, skutočnou revolúciou v tejto oblasti bol vynález biologického čipu (*microarray*).Predstava možnosti skúmať kód, ktorý riadi vývoj nášho organizmu nadchla mnoho vedcov po celom svete.

S novým prostriedkom pre výskum genómu však prišli aj nečakané problémy. Obrovské množstvo dát ktoré dokáže biologický čip poskytnúť o genóme nie je efektívne spracovávať štandardami metódami. Tu sa dostávajú k slovu metódy umelej inteligencie ktoré poskytujú množstvo efektívnych nástrojov pre analýzu a spracovanie takýchto dát. Jednou z takýchto metód je „Random Forests“ Lea Breimana, ktorá sa v tejto oblasti čoraz viac presadzuje.

1 Problematika dát získaných z čipov pre meranie génových expresií

Biologické čipy nazývané aj microarray sú technológia využívaná v molekulárnej biológií, medicíne a v iných príbuzných odvetviach, kde sa pracuje s genetickými informáciami. Je to moderná technológia, ktorej základy boli položené koncom osemdesiatych rokov. Pomocou čipov je v súčasnosti možné analyzovať štruktúru genómu a expresiu génov v relatívne krátkom čase. Ich hlavnou výhodou je schopnosť merania veľkého množstva génov (aj viac ako 200 000) v jednom experimente, na rozdiel od klasických molekulárnych technológií ako PCR alebo FISH.
Vedci neštudujú genómy iba preto, aby spoznali jednotlivé gény a ich evolúciu, ale aj preto, aby zistili, ako gény medzi sebou navzájom pôsobia, aby vytvárali a udržovali funkčný organizmus.
Technológia DNA čipov umožňuje práve takéto štúdie. Automatizácia dovoľuje ich uskutočňovanie v širokom rozsahu. Malé množstvo jednoreťazcových DNA fragmetov, ktoré predstavujú gény, je uložené na sklenenom sklíčku. Toto pole sa nazýva DNA čip kvôli jeho podobnosti s počítačovým čipom. V ideálnom prípade tieto fragmenty predstavujú celý genóm, to znamená všetky gény organizmu. To je možné u organizmov, ktorých genómy boli úplne sekvenované. Fragmenty sú testované hybridizáciou v rozličných vzorkách cDNA molekúl, ktoré boli označené fluorescenčnou značkou.

Z čipov sa získavajú dáta, ktoré predstavujú mieru expresií jednotlivých génov. Expresia génu je prejavenie sa génu navonok. Tieto dáta sú teda číselným vyjadrením miery expresie génu.
Dáta získané z čipov sa využívajú napríklad pre identifikáciu génov, ktoré spôsobujú genetické ochorenia.

1.1 Princíp DNA čipov

Princípom technológie čipov je nanesenie mikroskopických bodiek (spotov) určitých molekúl na pevný podklad ako napríklad sklo, plast, alebo silikón. Technológia DNA čipov bola predstavená v roku 1995, sú na čip „vytlačené“ tisícky variantov jednotlivých polymérov, ako DNA, RNA, alebo cDNA [6].

Každá varianta je zobrazená v malej bodke, známej ako sonda, alebo spot, obsahujúca kópie jednej a tej istej relatívne krátkej sekvencie polyméru. Látkové množstvo takýchto polymérov je v jednom spote v rádoch pikomólov (10^{-12} mol). Polymér v spote predstavuje jeden konkrétny gén DNA, ktorého expresia je v tomto spote skúmaná.

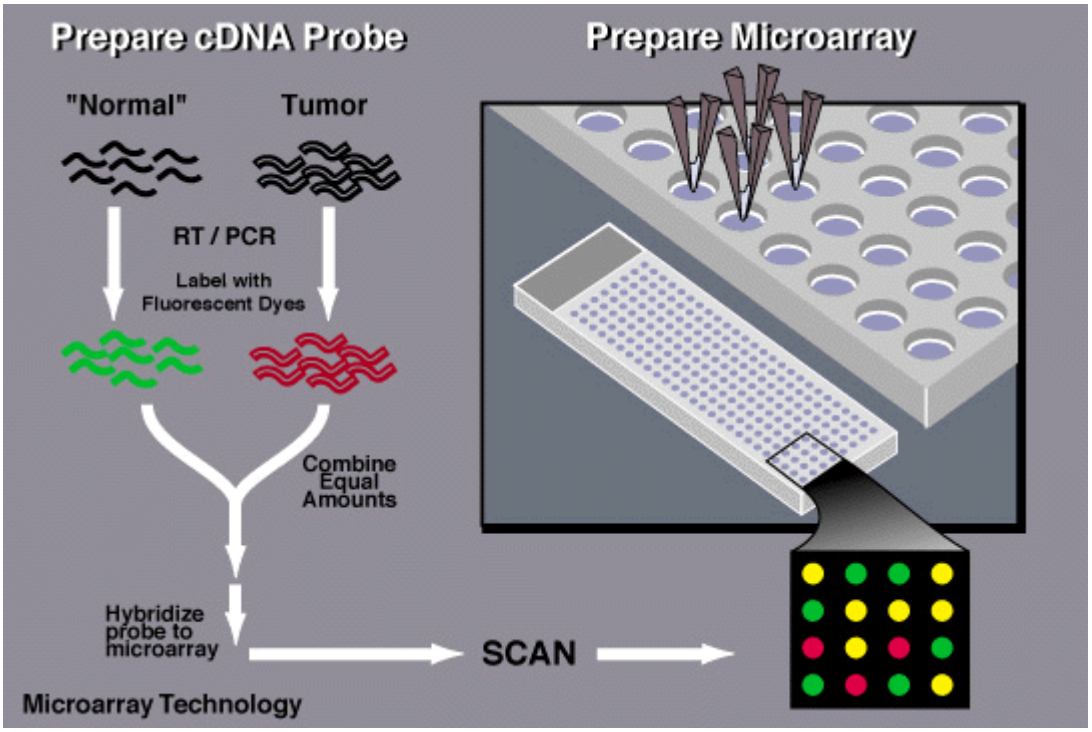
Expresia génu je proces pri ktorom sa informácia z génu použije na syntézu funkčného genetického produktu ako RNA alebo proteínu. Inak povedané, je to prejav génu.

Existuje niekoľko rôznych druhov DNA čipov pre rôzne druhy genetických experimentov. Podľa využitej čipovej analýzy a podľa typu sond na povrchu čipu možno rozdeliť DNA microarrays do dvoch skupín. Tzv. expresné čipy obsahujú ako sondy buď dvojreťazové úseky molekúl cDNA (komplementárna DNA) vytváraných reverznou transkripciou mRNA, alebo oligonukleotidové sondy sekvenčne špecifické pre každý gén z genómu [7].

Pre priblíženie princípu získania génových expresií pomocou DNA čipu si popíšeme proces využívaný pri cDNA čipoch. Proces získania génových expresií DNA pomocou cDNA čipu prebieha v piatich základných krokoch[8]:

1. Zo skúmanej DNA získame transkripciou mRNA
2. Spätnou transkripciou získame cDNA komplementárna ku skúmanej DNA
3. Takto získanú cDNA ofarbíme fluorescenčným farbivom aby ju bolo možné identifikovať
4. Ofarbenú cDNA prenesieme na DNA čip kde prebehne proces hybridizácie
5. Nakoniec z čipu odstránime prebytočnú cDNA a preskúma sa fluorescencia mikročipov.

Každý fluorescenčný bod predstavuje gén exprimovaný vo vzorke tkaniva. Cieľová, alebo skúmaná sekvencia, označená fluorescenčnými značkami je nanesená na čip. Ak sú body naviazané na cieľovú sekvenciu, buď sú zobrazené fluorescenčným farbivom, alebo sú detekované nejakou inou zobrazovacou metódou. Intenzita fluoreskovania každej sondy indikuje relatívnu kvantitu obsahu danej sekvencie v skúmanej vzorke.



Obrázok 1

V každom bode na mikroskopickom sklíčku sú upevnené kópie krátkej jednoreťazcovej molekuly DNA predstavujúcej jeden gén organizmu, rozdielny gén v každom bode.

Ale dôležitejšie, ako je potvrdenie tohto predpokladu, sú možnosti využitia metódy pri odhaľovaní nových génov, genetických interakcii a funkcií génov. Microarray analýzy sa využívajú napr. pri porovnávaní rakovinových a nerakovinových tkanív. Štúdium rozdielov v génovej expresií môže viesť vedcov k novým diagnostickým postupom a k biochemicky cielenej liečbe a taktiež k plnému pochopeniu tejto choroby. V podstate nám môžu informácie z čipových analýz poskytnúť obrovský rozhľad – ako súbory génov vzájomne spolupracujú a vytvárajú živý organizmus [4].

Veľkou výhodou čipov je, že každá genetická informácia je po zoskenovaní na špeciálnom zariadení kvantifikovateľná, a tým pádom je ju možné spracovávať na počítači a následne aj strojovo

analyzovať. Biologický čip je však schopný poskytnúť veľké množstvo údajov. Navyše zložité biologické laboratórne postupy, ktoré vedú k zobrazeniu genetickej informácie na čipe, nie sú vždy úplne exaktné. Preto je pri analýze takýchto dát potrebné zvoliť špecializovaný prístup, ktorý zohľadňuje práve tieto skutočnosti.

Súčasná experimentálna biológia sa čoraz viac zameriava na poznanie molekulárnej podstaty javov prebiehajúcich v živých systémoch. Požiadavky rýchlej a spoľahlivej diagnostiky veľkého počtu génov si vyžiadali nové prístupy v oblasti rozvoja biotechnológií. Vďaka technológii DNA čipov dnes dokážeme sledovať v jeden okamih expresnú aktivitu tisícov génov a pomocou vhodných nástrojov môžeme namerané hodnoty dávať do súvislostí, ktoré boli donedávna nemysliteľné.

1.2 Získavanie a spracovanie dát z DNA čipov

DNA čip s už hybridizovanou ofarbenou skúmanou vzorkou sa vkladá do špeciálneho skenera pre mikročipy. Skener je vybavený laserom, ktorý slúži k excitácií fluorescenčných farbív a snímacou optickou sústavou, určenou pre prácu s vlnovými dĺžkami ktoré emitujú farbivá na čipe. Laser osvieti ofarbenú vzorku (excitácia) a snímacia sústava zachytí evokované svetlo. Skenery obvykle umožňujú nastavenie nasledujúcich parametrov:

- intenzita laseru
- citlivosť snímania
- priestorové rozlíšenie

Nesprávne nastavenie intenzity laseru alebo citlivosti snímania má za následok šum v zosnímanom obraze čipu. Nesprávne zosnímané údaje môžu narušiť analýzu takto získaných dát. S týmto faktorom je potrebné počítať.

Pomocou vzorkovania je obraz čipu prevedený na diskrétne hodnoty. Pomocou rôznych špecializovaných algoritmov sa v diskrétnom obraze potláča náhodný šum a zvýrazňujú dôležité informácie ktoré sú dôležité. Takto upravený obraz sa potom prevádza na číselné hodnoty predstavujúce mieru expresie génu v každom jednom spote. Na základe intenzity pixlov sa vyčíslujú numerické hodnoty, ktoré musia byť následne ešte normalizované. Takto upravené dáta už slnujú požadované vlastnosti pre funkčnú analýzu. [10]

2 Metódy strojového učenia pre klasifikáciu dát z DNA čipov

Strojové učenie je podoblasťou umelej inteligencie, zaoberajúca sa algoritmami a technikami, ktoré umožňujú počítačovému systému "učiť sa". Učením v danom kontexte rozumieme takú zmenu vnútorného stavu systému, ktorá zefektívni schopnosť prispôsobenia sa zmenám okolitého prostredia. Strojové učenie sa značne prelína s oblasťami štatistiky a dobývaní poznatkov a má široké uplatnenie[11].

2.1 Diskriminačná analýza

Diskriminačné analýza je významnou metódou lineárneho modelovania. Používa sa buď pre interpretáciu rozdielov medzi definovanými skupinami objektov, alebo pre klasifikáciu, teda zaradenie nových objektov do tried.

Klasická klasifikačné diskriminačná analýza, zavedená Ronaldom Fisherom v roku 1936, patrí medzi metódy skúmania vzťahov medzi skupinou p nezávislých znakov a jednou kvantitatívnou závislou premennou. [12]

2.1.1 Princíp metódy LDA

Princípom klasifikácie pomocou Lineárnej diskriminačnej analýzy je hľadanie lineárnej kombinácie premenných (nazývaných aj diskriminantory), ktoré charakterizujú triedy objektov. Výsledná lineárna kombinácia je potom použitá ako lineárny klasifikátor.

Úlohou je nájdenie štatisticky najvýhodnejšieho postupu rozlíšenia medzi dvoma prípadne viacerými triedami, pričom pri každom prvku je známych niekoľko znakov (premenných) charakterizujúcich skúmané vlastnosti hodnotených jednotiek. Pri riešení klasickej diskriminačnej úlohy sú vopred známe skupiny prvkov a o každom prvku vieme, do ktorej skupiny patrí. Na základe merania jednotlivých prvkov v skupinách sa potom konštruuje tzv. diskriminačná funkcie, ktorá slúži k tomu, aby dodatočne uvažovaná jednotka, o ktorej nevieme, do ktorej skupiny patrí, mohla byť zaradená do správnej skupiny.

Pri diskriminačné analýze sa snažíme vyčísliť hodnotu diskriminačnej funkcie, ktorá nám uľahčí zaradenie do primárnej triedy. Takto vyčíslené hodnoty funkcie používame k triedeniu nezaradených objektov do vopred známych g tried a to na základe p diskriminátorov $x_1 , x_2 , ..., x_p$. Každá primárna trieda je charakterizovaná svojou funkciou hustoty pravdepodobnosti $f_j(x)$, kde $\boldsymbol{x} = [x_1 , x_2 , ..., x_p]$. Existuje citlivé pravidlo pre zaradenie, diskrimináciu objektu vektoru $\boldsymbol{x}$, do triedy G_j.

$$f_j(x) = max\ f_i(x)\ (i = 1..g)$$

Princípom LDA je nájdenie takej lineárnej kombinácie p sledovaných premenných aby lepšie ako akákoľvek iná lineárna kombinácia separoval uvažovaných G tried v tom zmysle, že ich vnútroskupinová variabilita bude čo najmenšia a medziskupinová variabilita bude čo najväčšia.

Metodický postup :

- Výpočet skupinových priemerov a matice rozptylu
- Diskriminácia pomocou rozhodovacieho pravidla
- Výpočet lineárnych diskriminačných funkcií
- Vyhľadanie najvyššej funkcie a s ňou súvisiacej pravdepodobnosti pre zaradenie každého jednotlivého pozorovania do príslušnej skupiny

Diskriminačná analýza vyžaduje predbežne určený počet skupín a ich konkrétnu definíciu.[12]

2.2 SVM

Metóda SVM (Support Vector Machines - metóda podporných vektorových strojov) bola vyvinutá ako lineárny klasifikátor a neskôr bola modifikovaná pomocou kernelových metód, takže umožňuje aj klasifikáciu lineárne neseparovateľných problémov. SVM rozdeľuje p-rozmerné dáta pomocou p-1 rozmernej separačnej hyperroviny[21].

Otázka je ako najlepšie umiestniť lineárnu hranicu tak aby bola čo najefektívnejšia z hľadiska kategorizácie budúcich dát, ktoré nebolo možné použiť pri tréningu. Podstatou metódy SVM je nájdenie takej optimálnej separačnej hyperroviny na základe trénovacích dát, ktorá správne rozdeľuje do tried aj vzorky z testovacej množiny, o ktorých predpokladáme, že majú rovnaké rozdelenie

pravdepodobnosti ako trénovacie dáta. Trénovacie vektory, ktoré sa nachádzajú najbližšie k separačnej hyperrovine, sú označované ako podporné vektory.

2.2.1 Klasifikácia lineárne separovateľných dát.

Majme L vstupných trénovacích vzoriek, kde každý vstup x_i má D atribútov (čiže je D-dimenzionálny) a patrí do jednej z dvoch tried y_i = -1 alebo +1. V tomto prípade predpokladáme, že tieto dáta sú lineárne separovateľné . Ak *x* je vektor hodnôt, potom separačná hyperrovina je potom definovaná ako:

$$w \cdot x + b = 0$$

kde w je normálový vektor oddeľujúci nadroviny, **b** je posunutie, pričom |**b**| / |**w**| je vzdialenosť od nadroviny k počiatku súradnej sústavy. Vzdialenosť medzi trénovacím vektorom **wi** a hyperrovinou nazývame rozpätie (margin). Možeme ho vyjadriť nasledovne:

$$\frac{|w \cdot x + b|}{\|w\|}$$

Vzťah ??? definuje nekonečne veľa predpisov tej istej roviny, teda ak **w** a **b** budú násobené rovnakou konštantou, zadefinujeme toto obmedzenie:

$$min|w \cdot x + b| = 1$$

Optimálna separačná hyperrovina maximalizuje rozpätie definované vzťahom ???, pričom je brané do úvahy obmedzenie dané vzťahom ???, je tento problém redukovaný na maximalizáciu výrazu, ktorý je možné formulovať ako:

$$\textit{minimalizácia } \mathbf{w} \textit{ vzhľadom na } \mathbf{y_i(w \cdot x_i + b) \geq 1}$$

kde y_i je 1 ak x_i patrí jednej triede a -1 v prípade ak x_i patrí do druhej z tried. Tento problém je možné zredukovať na problém riešiteľný metódou kvadratického programovania.[22]

minimalizácia

$$-\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j + \sum_i \alpha_i$$

vzhľadom na

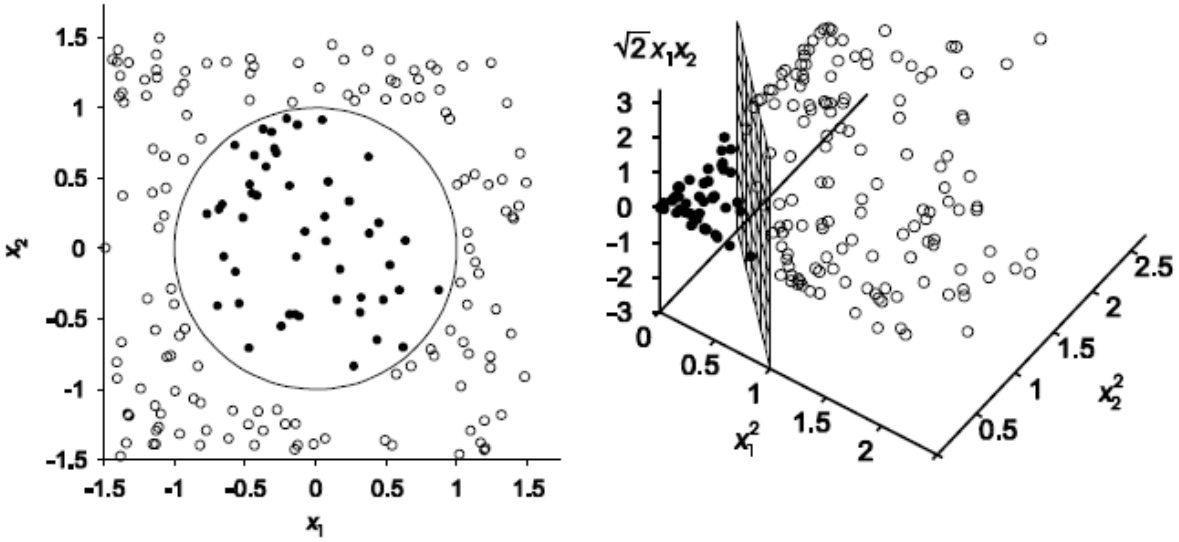
$$\sum_i \alpha_i y_i = 0, \ \ \alpha_i \geq 0$$

2.2.2 Klasifikácia lineárne neseparovateľných dát pomocou kernelových metód

Bežne nemožno očakávať že lineárny oddelovač bude možné nájsť priamo v originálnom vstupnom priestore. Takéto prípady vyžadujú nelineárny klasifikátor. Metóda SVM tento problém rieši pomocou takzvaných kernelových metód.

Základným princípom kernelových metód je transformácia dát do vysokorozmerného priestoru. Využitím vhodnej transformácie môžeme aj nelineárne separovateľný problém rozdeliť lineárnou hyperrovinou. Graficky je tento prístup znázornený na obrázku 2 vľavo, kde nie je možné použiť lineárny klasifikátor. Pridaním ďalšej dimenzie, vid obrázok 2 vpravo je možné jednotlivé triedy separovať lineárne. (Tento viacdimenzionálny priestor rozdeľujeme na dve polroviny lineárnym separátorom).

Pri mapovaní do priestoru s dostatočným počtom dimenzií môžeme nakoniec vždy nájsť lineárny oddeľovač (nadrovinu). Obecne platí, že **N** dátových bodov je možné vždy, okrem niektorých špeciálnych prípadov lineárne oddeliť v priestore s **N-1** alebo viac dimenziami.



Obrázok 2 [21]

Nech **Φ** je transformácia do vysokorozmerného priestoru. Vysokorozmerný priestor zachováva vzťah medzi vzdialenosťami vzoriek v pôvodnom a vysokorozmernom priestore. Definujme kernelovú funkciu K(x_i; x_j), pričom platí:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

kde K je kernelová funkcia, **Φ** je transformácia a x_i, x_j sú vektory v pôvodnom priestore. Zo vzťahu ??? vyplýva, že kernelová funkcia K je identická vzájomnej vzdialenosti vektorov x_i, x_j meranej vo vysoko-rozmernom priestore definovanom transformáciou **Φ**. Separačná hyperrovina vo vysoko-rozmernom priestore je definovaná nasledovne:

$$w \cdot \Phi(x) + b = 0$$

Optimalizačný problém vo vysokorozmernom priestore potom vyzerá nasledovne:

minimalizácia

$$-\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) + \sum_i \alpha_i$$

vzhľadom na

$$\sum_i \alpha_i y_i = 0, \;\; \alpha_i \geq 0$$

Zo vzťahu vyplýva, že pre optimalizáciu je možné použiť priamo kernelovú funkciu $K(x_i, x_j)$, pričom nie je nutné poznať ani počítať transformáciu **Φ**, čo je v literatúre uvádzané ako tzv. "kernel trick". Týmto je umožnené výpočtovo zvládnuť nelineárnu klasifikáciu.[21]

2.3 K-najbližších susedov

Metóda K-najbližších susedov (k-nearest neighbor, k-NN) je klasifikačná metóda pre učenie s učiteľom. Pre tento klasifikátor je každý prvok reprezentovaný *n*-ticou atribútov číselnej hodnoty. Jednotlivé vzorky sú teda obsiahnuté v *n*-dimenzionálnom priestore. Vo fáze učenia sa predspracuje trénovacia množina tak, aby všetky príznaky mali strednú hodnotu 0 a rozptyl 1. Takto sa prvky trénovacej množiny uložia do *n*-dimenzionálneho priestoru. Vo fáze klasifikácie sa prvok ktorý nie je zaradený v žiadnej triede zobrazí do natrénovaného priestoru a zistí sa jeho vzdialenosť od všetkých prvkov trénovacej množiny. Následne sa vyhľadá *k*-najbližších prvkov. Objekt je potom klasifikovaný do takej kategórie, do ktorej patrí väčšina z *k*-najbližších prvkov. Pre určenie vzdialenosti prvkov môžeme použiť rôzne metriky (Euklidovskú, Hammingovú, kosínovú, overlap metrika).

Metóda K-najbližších susedov je všeobecne použiteľná metóda klasifikácie s jednoduchou implementáciou, nízkymi výpočtovými nárokmi a vysokou účinnosťou. Je vhodná predovšetkým pre klasifikáciu do menšieho počtu tried .
V porovnaní s rozhodovacími stromami a niektorými druhmi neurónových sietí je však metóda k-NN menej stabilná.[12]

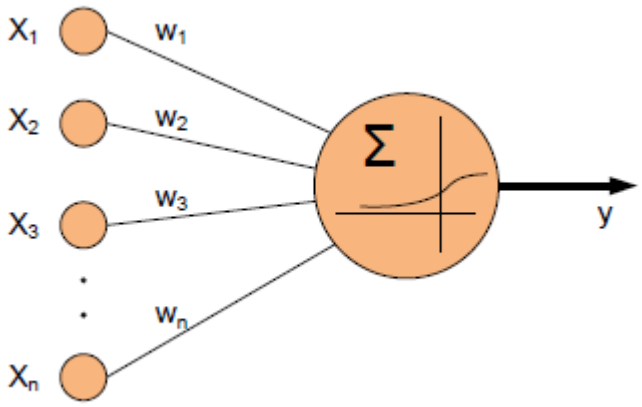
2.4 Neurónové siete

Metóda neurónových sietí vychádza z neurofyziologických poznatkov. Neurónové siete modelujú chovanie biologických štruktúr. Základnou jednotkou nervovej sústavy je neurón.
Skladá sa:

- telo neurónu – neurit
- vstupné výbežky – dendrity
- jedno výstupné vlákno – axón

Dendrity sa s axónami spájajú prostredníctvom tzv. synapsií. V synapsiách sa uvoľňujú chemické látky, ktoré ovplyvňujú prenos elektrických impulzov. Takýmto spôsobom sú signáli prenášané v neurónovej sieti. Chemické látky môžu pôsobiť ako excitátory alebo inhibítory. Excitátory zvyšujú a inhibítory naopak znižujú schopnosť nasledujúceho neurónu generovať impulz. Hodnoty vstupných signálov sa v tele neurónu sčítajú. Ak ich súčet presiahne určitý prah, neuron vyšle krátky impulz po axóne a tým môže aktivovať ďalšie neuróny.[24]

Na princípe činnosti biologického neurónu bol zavedený umelý neurón. Umelý neurón nazývaný aj preceptrón bol predstavený Frankom Rosenblattom v roku 1958. Schéma perceptrónu je zobrazená na obrázku 3.



Obrázok 3 [23]

2.4.1 Umelý neurón

Perceptrón [24] má niekoľko vstupov a práve jeden výstup. Vstupný vektor ($x_1,...,x_n$) modeluje vstupy, čiže dendridy biologického neurónu. Každému vstupu je priradená váha ($w_1,...,w_n$). Tieto váhy udávajú mieru dôležitosti jednotlivých vstupov. Potenciál neurónu je potom daný váženou sumou vstupov a prahom neurónu označeným Θ. Tento vnútorný potenciál vstupuje do aktivačnej funkcie *S(x)* ktorej výstupná hodnota je zároveň aj výstupom neurónu Y.
Výstup neurónu je teda definovaný rovnicou :

$$Y = S\left(\sum_{i=1}^{N}(x_i w_i) - \Theta\right)$$

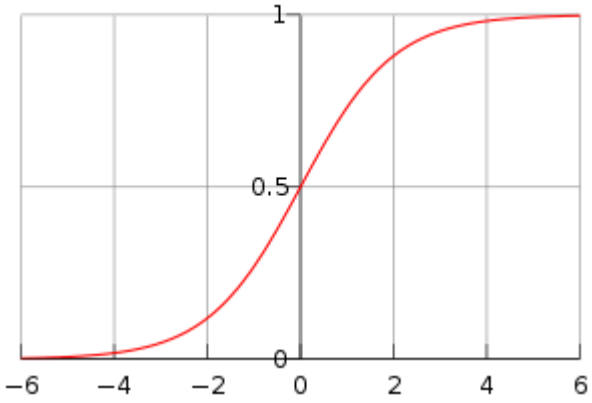
Samotná aktivačná funkcia môže mať rôzny tvar. V praxi sa používa niekoľko typov.
Najjednoduchším typom je skoková funkcia, ktorá môže byť popísaná vzťahom:

$$S(x)\begin{cases}0\ pre\ x < 0\\1\ pre\ x > 0\end{cases},\quad kde\ \sum_{i=1}^{N}(x_i w_i) - \Theta$$

Najčastejšie používaným typom aktivačnej funkcie je sigmoidálna funkcia. Jej tvar je vyjadrený vo vzťahu:

$$S(x) = \frac{1}{1 + e^x}$$

Priebeh sigmoidálnej funkcie je zobrazený na obrázku 4.



Obrázok 4

Medzi ďalšie často používané aktivačné funkcie patrí napríklad hyperbolická tangenta alebo aj jednoduchá lineárna funkcia.

2.4.2 Umelá neurónová sieť

Perceptrón je základnou jednotkou neurónovej siete. Poprepájaním perceptrónov vzniká neurónová sieť. Najjednoduchšia neurónová sieť sa skladá z jedného perceptrónu. Podľa počtu preceptronov, typu prepojenia a typu aktivačnej funkcie rozoznávame niekoľko druhov typov neurónových sietí:

- Dopredná neurónová sieť
- Hopfirldova neurónová sieť
- Rekurentná neurónová sieť

Najznámejšou a najrozšírenejšou metódou pre učenie dopredných neurónových sietí je takzvaná BackPropagation. Učenie siete metódou BackPropagation je založené na spetnom prechode sieťou a minimalizáciou celkovej chyby pre všetky vzory trénovacej množiny. Chybou vzorky rozumieme odchýlku výstupných hodnôt vzhľadom na požadované. Na základe tejto odchýlky sú postupne upravované váhy od vstupnej až po výstupnú vrstvu.

Takáto sieť má najmenej tri vrstvy. Jednu vstupnú vrstvu, na ktorú sú privádzané vstupné dáta, jednu výstupnú vrstvu. Hodnoty neurónov vo výstupnej vrstve predstavujú výsledok klasifikácie, čiže triedu do ktorej bola vstupná vzorka klasifikovaná. Medzi vstupnov a výstupnov vrstvou sa nachádza jedna, alebo viacero skrytých vrstiev. Pre vytvorenie kvalitného klasifikátora často stačí aj jediná skrytá vrstva.

Pre metódu BackPropagation je potrebné aby aktivačná funkcia neurónov bola diferencovateľná. Túto podmienku splňuje napríklad sigmoidálna funkcia:

$$\frac{dS}{dx} = S(x) = S(x)(1 - S(x))$$

2.5 Rozhodovacie (klasifikačné) stromy

Rozhodovací strom je graf stromovej štruktúry, kde každý vnútorný uzol reprezentuje test hodnoty istého atribútu a koncové uzly (listy) reprezentujú triedu, do ktorej je daný objekt klasifikovaný. Takýto rozhodovací strom môže byť ľahko prevedený na zodpovedajúce klasifikačné pravidlá. Klasifikačný strom predstavuje model pre dáta, kde každé pozorovanie patrí do niektorej z tried C_1,. . . , C_k, $k \geq 2$. Súčasne je pozorovanie charakterizované vektorom $x = (x_1,..., x_m)$ hodnôt vysvetľujúcich premenných (prediktor) X_1,. . . , X_m. V jednej a tej istej úlohe sa môžu vyskytovať prediktory kvantitatívne aj kvalitatívne.

V každom neterminálnom uzle sa strom vetví - z uzla vedú hrany do dvoch alebo (v niektorých metódach) aj viacerých dcérskych uzlov. Vetvenie je založené na hodnote jedného prediktoru. Najbežnejšie je binárne vetvenie podľa odpovede na otázku tvaru „x_i <c?“ pre kvantitatívne prediktory X_i a „$x_i \in B$?“ (kde B je neprázdna vlastná podmnožina množiny všetkých hodnôt veličiny X_i) pre prediktory X_i kvalitatívne (kategorické). Jedna hrana je potom priradená kladnej a druhá zápornej odpovedi. (Niektoré metódy umožňujú aj vetvenie založené na lineárnej kombinácii kvantitatívnych prediktorov.)

Pozorovanie podľa hodnôt prediktoru „postupuje“ od koreňového uzla cez vetvenia v neterminálnych uzloch k niektorému terminálnemu uzlu (listu). Množina všetkých listov určuje disjunktnivný rozklad priestoru hodnôt prediktoru X. Terminálnímu uzlu a zároveň pozorovaniam, ktorá do neho patria, je priradená niektorá z tried C_1,. . . , C_k. Strom T tak určuje klasifikačná funkciu d_T definovanú na X s hodnotami v množine (C_1,. . . , C_k).

K vytváraniu (pestovaniu) stromov prakticky všetky bežné metódy využívajú tzv. rekurzívne delenie (recursive partitioning). Konštrukcia začína stromom o jednom uzle, do ktorého patria všetky trénovacie dáta (koreň je zároveň listom). Preberie sa množina všetkých možných vetvení a pre každé z nich sa vypočíta kriteriálna štatistika (splitting criterion), ktorá - nech je povedané bez podrobností - hodnotí, nakoľko sú potenciálne dcérske uzly hodnotami závislé premenné vnútorne homogénne a navzájom odlišné. Vetvenie s maximálnou hodnotou kritéria sa vyberie ako najlepšie a uplatní sa v modeli, ku ktorému tak pribudne dvojica (popr. v niektorých metódach väčší počet) uzlov, ktoré sú zatiaľ terminálne. Dáta, ktoré patria do koreňového uzla, sa rozdelia podľa hodnôt prediktorov medzi nové dcérske uzly. Pre každý z týchto provizórnych listov sa procedúra opakuje, ako keby sa jednalo o koreň - hľadá sa najlepšie vetvenie, atď. [2]

2.5.1 Bagging

Bagging je skratka „bootstrap aggregating“. Z trénovacieho dátového súboru $L = ((x_1, y_1),. . . , (x_n, y_n))$ sa vytvorí náhodným výberom s vrátením L súborov $L_1,. . . , L_L$ veľkosti n (bootstrapových výberov), každý z nich sa použije na zostrojenie jedného klasifikačného stromu a výsledný klasifikačný les je potom daný väčšinovým hlasovaním s rovnakými váhami (resp. aritmetickým priemerom čiastkových regresných funkcií).
Do bootstrapového výberu sú niektoré pozorovania z *L* vybrané opakovane a niektoré naopak vôbec. Počet opakovaní má pre jednotlivé pozorovania z *L* asymptotickej (pre $n \rightarrow \infty$) Poissonovo rozdelenie so strednou hodnotou 1. Pravdepodobnosť, že pozorovanie nebude vôbec vybrané, je teda približne $e^{-1} \approx 0.37$. Bootstrapový výber je teda tvorený asi 63% pozorovania z *L*, 37% zostáva mimo [2].
Tvorbu lesu pomocou bootsrapových výberov využíva aj metóda Random Forests.

2.5.2 Boosting a arcing

Boosting (to boost - zosilňovať) je pôvodne pojem z teórie strojového učenia a v analýze dát sa tak bežne označuje algoritmus AdaBoost.
Majme klasifikačnú metódu (nemusí sa jednať len o stromy), ktorá vytvára klasifikačný model T na základe trénovacích dát $(x_1, y_1),. . . , (x_n, y_n)$ a vektora $w = (w_1,..., w_n)$ váh priradených jednotlivým pozorovaním. Algoritmus AdaBoost konštruuje postupnosť rozdielnych modelov $T_1,. . . , T_L$ s klasifikačnými funkciami $d_1 (x),. . . , D_L (x)$ tak, že sa podľa predchádzajúcich výsledkov postupne upravujú váhy prípadov. V prvom kroku sa použije váhový vektor w_1 zadaný používateľom (napr. rovnomerné váhy) a vytvorí sa model T_1. V ďalších krokoch sa potom vždy ku konštrukcii modelu T_i $(i = 2,..., L)$ uplatňuje váhový vektor w_i získaný takouto úpravou vektora w_{i-1}, že sa váhy pozorovania chybne klasifikovaných modelom T_{i-1} zvýšia a správne klasifikovaných znížia. Klasifikačná metóda tak čoraz viac „sústreďuje pozornosť“ na „obtiažne“ pozorovania, ktoré vzdorujú zaradenie do správnej triedy. Váha modelu pri hlasovaní závisí na chybe modelu na trénovacích dátach. [2]

2.6 Random Forests

Random Forests je klasifikačná metóda navrhnutá Leom Breimanom. Metódu popisuje v článku [1] z ktorého som čerpal. Metódu popisuje nasledovne.
Random Forests (Náhodné lesy) vytvárajú mnoho klasifikačných stromov. Pre klasifikáciu nového objektu zo vstupného vektora, prejde vstupný vektor každým stromom v lese (skupine stromov). Každý zo stromov poskytne klasifikáciu. Hovoríme, že strom „hlasoval“ za triedu. Les potom vyberie klasifikáciu, ktorá má najviac hlasov zo všetkých stromov v lese.

Každý strom rastie nasledovne:

- Počet trénovacích prípadov označíme N a počet premenných v klasifikátore M.
- *m* vstupných premenných sa použije na určenie rozhodovania v uzle stromu, pričom platí že *m*<<M
- Vyberieme trénovaciu sadu pre tento strom N-krát nahradením zo všetkých N dostupných trénovacích prípadov. Zvyšok prípadov sa použije na určenie chyby stromu, pomocou predikcie ich tried.
- Pre každý uzol stromu sa náhodne vyberie *m* premenných, podľa ktorých bude uzol rozhodovať. Na základe týchto *m* premenných sa vypočíta najlepšie rozdelenie v trénovacej sade.
- Každý strom rastie do plnej veľkosti. Neorezáva sa.

V originálnej práci Lea Breimana o Random Forests bolo ukázané, že miera chyby lesa závisí na dvoch veciach:

- *Korelácia* medzi ktorýmikoľvek dvoma stromami v lese. Zvyšovaním korelácie sa zvyšuje aj miera chyby lesa.
- *Sila* každého jedného stromu v lese. Strom s nízkou mierou chyby je silným klasifikátorom. Zvyšovaním sily konkrétneho stromu sa znižuje miera chyby lesa.

Znižovaním *m* znižujeme oboje, koreláciu aj silu. Zvyšovaním naopak zvýšime. Niekde medzi tým je „optimálny“ rozsah *m* – často veľmi široký. Použitím OOB miery chyby (vysvetlene nižšie), môžeme rýchlo nájsť hodnotu *m* z rozsahu. Toto je jediný nastaviteľný parameter na ktorý je Random Forests citlivý [1].

2.6.1 Vlastnosti Random Forests

Podľa Breimana má Random Forests tieto vlastnosti:

- Nie je možné zlepšiť jeho presnosť v porovnaní s terajšími algoritmami,
- Je efektívny na veľkých databázach.,
- Môže spracovávať tisíce vstupných premenných bez mazania premenných,
- Poskytuje odhad, ktoré premenné sú dôležité pri klasifikácií,
- Generuje vnútorne podobné odhady v generalizácií chyby počas priebehu tvorby stromu,
- Obsahuje efektívnu metódu pre odhad chýbajúcich dát a udržiava presnosť ak veľké množstvo dát chýba,
- Obsahuje metódu pre vyváženie chyby v nevyvážených sadách dát triedy populácie,
- Generované stromy môžu byť uložené pre budúce použitie na iných dátach,
- Prototypy sú počítané tak, aby poskytovali informácie o vzťahoch medzi premennými a klasifikátormi,
- Vypočítava blízkosti (najbližšie) medzi pármi prípadov, ktoré môžu byť použité v klastrovaní, hľadaní vzdialených hodnôt, alebo (škálovaním) poskytnúť zaujímavé pohľady na dáta,
- Vyššie uvedené schopnosti môžu byť rozšírené na neoznačené dáta, čo vedie k neriadenému klastrovaniu, zobrazeniam dát, a odhalenia vzdialených hodnôt,
- Ponúka experimentálne metódy pre detekciu interakcií premenných.

Random Forests nevedie k preučeniu (overfitting). Je rýchly, aj keď beží na veľkom počte stromov. Ak máme 50 000 prípadov na 100 premenných, vyprodukuje 100 stromov za 11 minút na procesore s rýchlosťou 800Mhz. Pre veľké objemy dát je hlavnou požiadavkou na pamäť uloženie samotných

dát a tri polia integerov s rovnakými rozmermi ako dáta. Ak sú vzdialenosti vypočítané, pamäťové nároky narastajú s počtami prípadov krát počet stromov [1].

2.6.2 Ako Random Forests pracuje

Pre pochopenie a použitie rôznej možnosti nastavenia je užitočné vedieť ako sú počítané. Väčšina nastavení závisí na dvoch dátových objektoch generovaných pomocou Random Forests.

OOB (out-of-bag) : Keď trénovacia sada pre aktuálny strom je vytvorená výberom s opakovaním, približne jedna tretina prípadov sú nechané mimo vzorku. Tieto dáta, ktoré sú mimo (oob dáta) sú použité pre nezávislý odhad chyby klasifikácie, ako sú stromy pridávané do lesa. Taktiež sa používajú pre získanie odhadov dôležitosti premenných.

Proximities : Potom ako je už strom vytvorený, prebehnú ním všetky dáta, a pre každý pár prípadov sú vypočítané proximity, alebo inak povedané podobnosti. Ak sa dva prípady nachádzajú na jednom koncovom uzle stromu, naznačuje to ich podobnosť, ich proximita sa zvýši o jedna. Na konci behu sú proximity normované, a to tak, že sa vydelia počtom stromov. Proximity teda tvoria maticu rozmeru NxN, pričom N je počet vzoriek v trénovacej množine. Proximity sa používajú na nahradzovanie chýbajúcich dát, pri hľadaní vzdialených hodnôt a na vytváranie nízkodimenzionálnych pohľadov na dáta.

2.6.3 OOB pre odhad chyby

V Random Forests nie je potrebná krížová validácia, alebo samostatná testovacia sada pre získanie nezávislých odhadov chyby . Tento odhad je vytvorený okamžite, už počas behu a to nasledovne:

Každý strom je vytvorený za pomoci rôznych bootstrapových výberov z pôvodných dát. Približne jedna tretina prípadov sa nechá mimo bootstrapových výberov a nie sú použité pri konštrukcií *k*-teho stromu.

Pre získanie klasifikácie, každý prípad vynechaný pri konštrukcii *k*-teho stromu prejde cez *k*-ty strom. Takto sa získa klasifikácia testovacej sady pre každý prípad na približne tretine stromov. Na konci behu vezmeme *j* ako triedu, ktorá mala najviac hlasov vždy vtedy, keď bol prípad *n* OOB. OOB odhad chyby je pomer počtu ráz, kedy sa *j* nerovná skutočnej triede *n*, k priemeru všetkých tried. Mnohé testy dokázali, že takto je to nezávislé.

2.6.4 Význam premenných

Metóda dokáže na základe vytvoreného klasifikátora určiť význam jednotlivých premenných pre klasifikáciu. Cez každý strom, ktorý je vytvorený, necháme prejsť OOB prípady a spočítame výsledky, ktoré hlasovali za správnu triedu. Potom náhodne obmieňame hodnoty premennej *m* v OOB prípadoch a nechávame tieto prípady prejsť cez strom. Počet hlasov za správnu triedu v premennej-*m*-obmenenej OOB dátami odčítame od počtu hlasov za správnu triedu v zatiaľ nepoužitých OOB dátach. Táto hodnota môže teda byť aj negatívna, v prípade ak mala premenná negatívny vplyv na správnu klasifikáciu. Priemer týchto hodnôt nad všetkými stromami je prvotnou hodnotou významnosti premennej *m*.

Ak sú tieto hodnoty nezávislé od stromu k stromu, potom štandardná chyba môže byť vypočítaná pomocou štandardného výpočtu. Korelácie týchto hodnôt medzi stromami boli vypočítané na počet dátových setov a overené, aby boli čo najnižšie.

Ak je počet premenných privysoký, lesy môžu raz bežať so všetkými premennými a potom bežať zas, použitím iba najdôležitejších premenných z prvého behu.

Pre každý prípad, posúdime všetky stromy, pre ktoré je to OOB. Odčítame percento hlasov pre správne triedy v premennej-m-obmenenej OOB dátami od percenta hlasov hlasujúcich za správnu triedu v zatiaľ nepoužitých OOB dátach. Toto je hodnota lokálnej významnosti pre premennú *m* v tomto prípade a je použitá v grafickom programe RAFT (RAndom Forest Tool) [1].

2.6.5 Význam Gini

Gini koeficient je miera štatistického rozptýlenia. Vždy keď je rozdelením vytvorený nový uzol na premennej *m* je gini hodnota pre dva nasledujúce uzly nižšia ako u rodičovského uzla. Pridaním Gini pre každú individuálnu premennú na všetkých stromoch v lese poskytuje rýchle hodnoty významnosti premenných ktoré sú často veľmi podobné vypočítanej významnosti premenných.

2.6.6 Odľahlé objekty

Odľahlé objekty sú vo všeobecnosti objekty ktoré sa hodnotami svojich premenných výrazne odlišujú od ostatných. V prípade Random Forests môžeme povedať, že sú to objekty, ktorých proximity voči ostatným objektom sú vo všeobecnosti malé. Odľahlé objekty je vhodné definovať vzhľadom na ich triedu. V takom prípade odľahlý objekt triedy chápeme ako objekt ktorého proximity vzhľadom na ostatné objekty tej istej triedy sú nízke.

Priemernú proximitu objektu *n* v triede *j* definujeme nasledovne:

$$\bar{P}(n) = \sum_{class(k)=j} prox^2(n,k)$$

Kde funkcia *class(k)* určuje príslušnosť objektu do triedy a *prox(n,k)* určuje proximity hodnotu objektu *n* voči objektu *k*.
Odľahlosť objektu *n* je potom definovaná takto:

$$\frac{N}{\bar{P}(n)}$$

Kde N je celkový počet objektov.Táto hodnota bude vysoká ak bude priemerná proximita malá.

2.6.7 Nahradenie chýbajúcich hodnôt

Random Forests umožňuje dve metódy nahradzovania chýbajúcich hodnôt. Prvá metóda je rýchla. Ak *m*-tá premenná nie je kategorická, metóda vypočíta medián hodnôt tejto premennej v triede *j* a túto hodnotu použije pre nahradenie všetkých chýbajúcich premenných *m* v triede *j*. Ak je m-tá premenná kategorická, chýbajúce hodnoty sú nahradené najčastejšie vyskytujúcimi sa hodnotami tejto premennej v triede *j*.

Druhá metóda je výpočtovo náročnejšia ale poskytuje lepšie výsledky ako prvá metóda, aj v prípade veľkého počtu chýbajúcich dát. Metóda najprv vyplní chýbajúce hodnoty nepresnými hodnotami. Potom vypočíta proximity.

Ak je chýbajúca premenná nekategorická, nahradí sa priemernou hodnotou nechýbajúcich hodnôt premennej váhovaných proximitami medzi objektom s chýbajúcou hodnotou a objektom s nechýbajúcou hodnotou. Ak chýba kategorická premenná, je nahradená najčastejšie sa vyskytujúcimi hodnotami, ktoré sú takisto váhované proximitami.

2.7 Vyhodnocovanie kvality klasifikácie

Hodnotenie kvality metódy nezávislé na aplikačnej oblasti sa opiera predovšetkým o rôzne numerické parametre. Je ale potrebné zdôrazniť, že nie všetko, čo je v dátach presvedčivo dokázané, má pri posudzovaní kvality význam.

Pri hľadaní znalostí pre potreby klasifikácie sa obvykle postupuje metódou učenia s učiteľom. Vychádza sa teda z toho, že sú k dispozícií trénovacie dáta o ktorých vieme do akej triedy patria. Metódy hodnotenia sú potom založené na testovaní nájdených vedomostí na dátach, na možnosti porovnať, ako dobre sa nájdené znalosti zhodujú s informáciami od učiteľa. Pre testovanie sa ponúka celá rada variantov podľa toho, aké dáta použijeme pre učenie a aká pre testovanie:

- testovanie v celých trénovacích dátach
- krížová validácia (cross-validation)
- leave-one-out
- Bootstrap
- testovanie na testovacích dátach

Testovanie na celých dátach použitých pre trénovanie má najmänšiu výpovedajúcu schopnosť o tom, ako budú nájdené znalosti použiteľné pre klasifikáciu nových prípadov. Často totiž môže dôjsť k preučeniu (overfitting), kedy nájdené znalosti vystihujú skôr náhodné charakteristiky trénovacích dát a neodhaľujú to podstatné, čo možno použiť pre generalizáciu. Trénovacie dáta teda nie sú príliš vhodné pre testovanie nájdených znalostí a preto sa obvykle používajú iné dáta. Otázkou je, ako také dáta získať. Jedným problémom môže byť, že dát je k dispozícií málo, iný zas, že je žiaduce, aby dáta použité pre testovanie sa podobali dátam trénovacím. Oba tieto problémy možno riešiť rôznymi spôsobmi výberu trénovacích a testovacích dát.

Pri testovaní metódou krížovej validácie sa dáta dopredu rozdelia na napr. 10 častí tak, že vždy jedna desatina odoberie pre testovanie a zvyšných deväť desatín sa použije pre učenie. Celý tento postup sa zopakuje desaťkrát, vždy s inou desatinou a výsledok testovania sa spriemeruje. Tento konkrétny postup sa nazýva desatinová krížová validácia (10 fold cross-validation).

Variantov tohto prístupu je metóda leave-one-out. Z dát ktoré sú k dispozícií sa vyberie jeden príklad pre testovanie a zvyšné dáta sa použijú pre učenie. Toto sa opakuje toľkokrát, koľko príkladov máme k dispozícií. Vznikne teda *n* súborov znalostí, ktoré sa testujú na *n* príkladoch. Výsledok testovania dáva odhad, ako by sa znalosti získané zo všetkých dostupných *n* príkladov chovali pri klasifikovaní príkladov neznámych. V prípade že počet prípadov *n* je veľký (tisíce, desaťtisíce) je výhodnejšie použiť pomerovú krížovú validáciu, alebo je možné test leave-one-out implementovať efektívnejšie pomocou algoritmov na to určených (ESOD) [12].

Princíp metódy bootstrap je popísaný v časti Random Forests. Kedže v bootstrapovom výbere sa v priemere nachádza 63,2% zvyšných približne 36% je možné použiť pre testovanie klasifikátora. V algoritme Random Forests táto validácia prebieha priamo počas trénovania.

2.7.1 Matica zámen

Cieľom testovania je určiť, v koľkých prípadoch sa klasifikátor zhoduje s učiteľom a v koľkých prípadoch sa dopustil chyby. Tieto údaje sa zachytávajú v **matici zámen** [12] (*confusion matrix*). V matici zámen sú v stĺpcoch uvedené informácie o tom ako postupoval pri klasifikácií systém využívajúci nájdené znalosti. V riadkoch je uvedené správne(podľa učiteľa) zaradenie do triedy.

	Klasifikácia systémom	
Správne zaradenie	+	-
+	TP	FN
-	FP	TN

Tabuľka 1

Tabuľka 1 zachytáva situáciu kedy ide o klasifikáciu do dvoch tried „+“ a „-“. TP (správne pozitívny, true positive) je počet príkladov ktoré systém správne zaradil do triedy „+“, FP (falošne pozitívny, false positive) je počet príkladov, ktoré systém chybne zaradil do triedy „+“ (správne patria do triedy „-“). TN (správne negatívne, true negative) je počet príkladov ktoré systém správne zaradil do triedy „-“, FN (falošne negatívny, false negative) je počet príkladov, ktoré systém chybne zaradil do triedy „-“(správne patria do triedy „+“).

Matica zámen sleduje iba počty správne a nesprávne zaradených príkladov. V mnohých prípadoch môže byť dôležitý aj typ chyby, ktorého sa systém dopustil (FP vs. FN). Pri hodnotení vedomostí sa teda nemusí brať do úvahy prostý počet chybných rozhodnutí, ale aj cena takýchto chýb.

2.7.2 Celková správnosť

Celková správnosť [12] (overall accuracy) – resp. úspešnosť(succesfulness), alebo komplementárna celková chyba (overall error) sú najednoduchšími charakteristikami toho, ako sú získané znalosti kvalitné.
Celková správnosť sa počíta ako relatívny počet správnych rozhodnutí systému:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Celková chyba sa spočíta ako relatívny počet chybných rozhodnutí:

$$Err = \frac{FP + FN}{TP + TN + FP + FN}$$

Celková správnosť by mala byť z intervalu (Acc_{def} , Acc_{max}), kde Acc_{def} je správnosť systému, ktorý všetky príklady priradí k majoritnej triede a Acc_{max} je maximálna správnosť dosiahnutá pre dané dáta.

V prípade že nás pri testovaní zaujíma iba počet správnych resp. nesprávnych rozhodnutí systému:

$$Err = 1 - Acc$$

2.7.3 Správnosť pre jednotlivé triedy

V prípade, že triedy sú v dátach rozložené výrazne nerovnomerne, bude celková správnosť dávať skreslený obraz o nájdených znalostiach. 95% správnosť môže znamenať, že sme rozpoznali iba všetky prvky v triede „+“ alebo žiadne prvky v triede „-“. V takomto prípade je vhodnejšie sledovať správnosť pre jednotlivé triedy:

$$Acc_{+} = \frac{TP}{TP + FP}$$

$$Acc_{-} = \frac{TN}{TN + FN}$$

2.7.4 Presnosť a úplnosť

Presnosť a úplnosť [12] (precision and recall) sú pojmy používané v oblasti vyhľadávania informácií. Ak hľadáme napríklad na internete nejaké dokumenty týkajúce sa určitej témy pomocou vyhľadávača, potom:

1. nie všetky nájdené dokumenty sa týkajú nami hľadanej témy
2. určite nebolo nájdené všetko čo je o téme k dispozícií

Presnosť nám hovorí, koľko nájdených dokumentov sa skutočne týka danej témy a úplnosť nám hovorí, koľko dokumentov týkajúcich sa téme sme našli. Tieto miery zhody možno použiť aj na hodnotenie znalostí:

$$\text{Presnosť} = \frac{TP}{TP + FP}$$

$$\text{Úplnosť} = \frac{TP}{TP + FN}$$

Ako je vidieť, presnosť je to isté ako správnosť pre danú triedu. Niekedy sa používa súhrnná charakteristika, F-*miera*.

$$\text{F} = \frac{2 * \text{Presnosť} + \text{Úplnosť}}{\text{Presnosť} + \text{Úplnosť}} = \frac{2TP}{2TP + FP + FN}$$

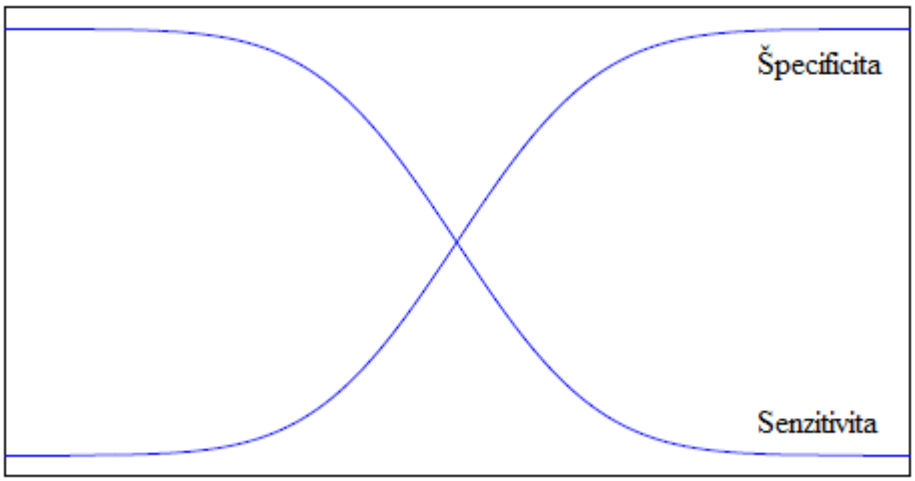
2.7.5 Senzitivita a špecificita

Senzitivita a špecificita [12] (sensitivity a specificity) sú charakteristiky prevzaté z medicíny.
V prípade nasadenia nejakého nového lieku nás zaujíma, pri koľkých chorých pacientoch liek zaberie (senzitivita), a či liek zaberá iba na danú (chorobu špecificita). Z matice zámen sa tieto hodnoty spočítajú nasledovne:

$$\text{Senzitivita} = \frac{TP}{TP + FN}$$

$$\text{Špecificita} = \frac{TN}{TN + FP}$$

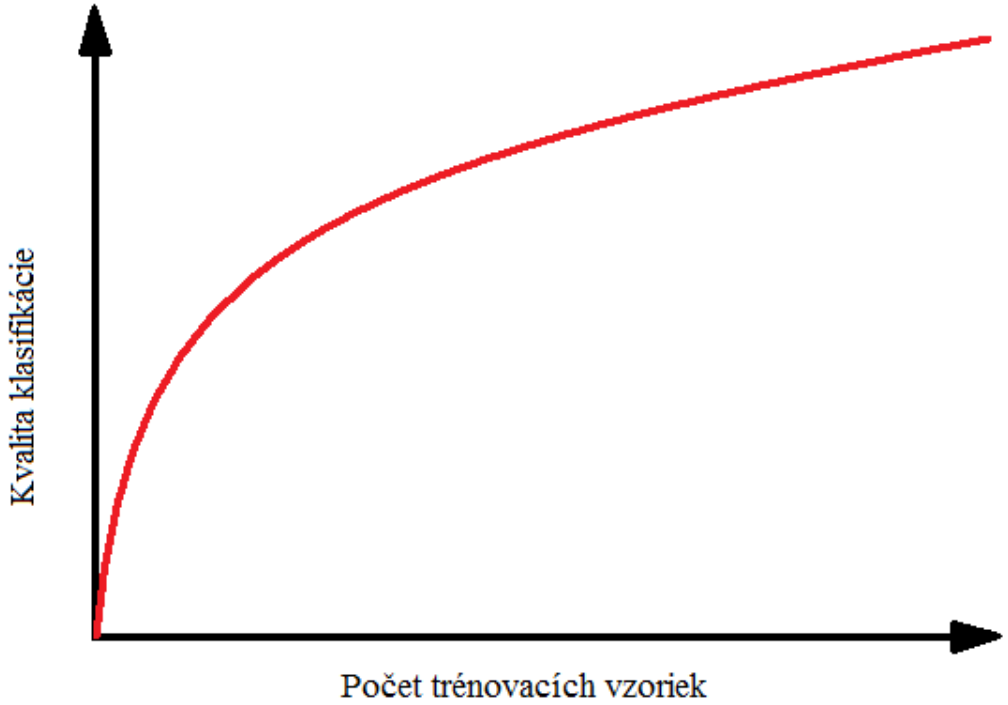
A opäť môžeme pozorovať zhodu používaných kritérií; senzitivita je to isté čo úplnosť. Senzitivita a špecificita sa buď uvádzajú samostatne alebo ako vzájomný súčin oboch čísel.



Obrázok 5

2.7.6 Krivka učenia

Krivka učenia (learning curve) dáva do súvislosti počet príkladov v trénovacej množine a správnosť klasifikácie (Obr. 3).



Obrázok 6

Vychádza sa z predpokladu, že čím viac príkladov je k dispozícii vo fáze učenia, tým budú nájdené znalosti presnejšie. Pre rôzne počty príkladov teda dostaneme rôzne hodnoty správnosti pri testovaní.

Testovanie sa často robí opakovane, takže okrem priemernej hodnoty správnosti $\overline{Acc}$, získame ešte jej chybu. Táto chyba sa obvykle počíta ako

$$\frac{S_{Acc}}{\sqrt{n}}, \text{ kde } S_{Acc}^2 = \frac{\sum_i \left(Acc_i - \overline{Acc}\right)^2}{n-1}$$

Istou analógiou tohto prístupu je krivka učenia používaná pri učení neurónových sietí ako indikácia toho, kedy je možné učenie ukončiť. [12]

2.7.7 Krivka ROC

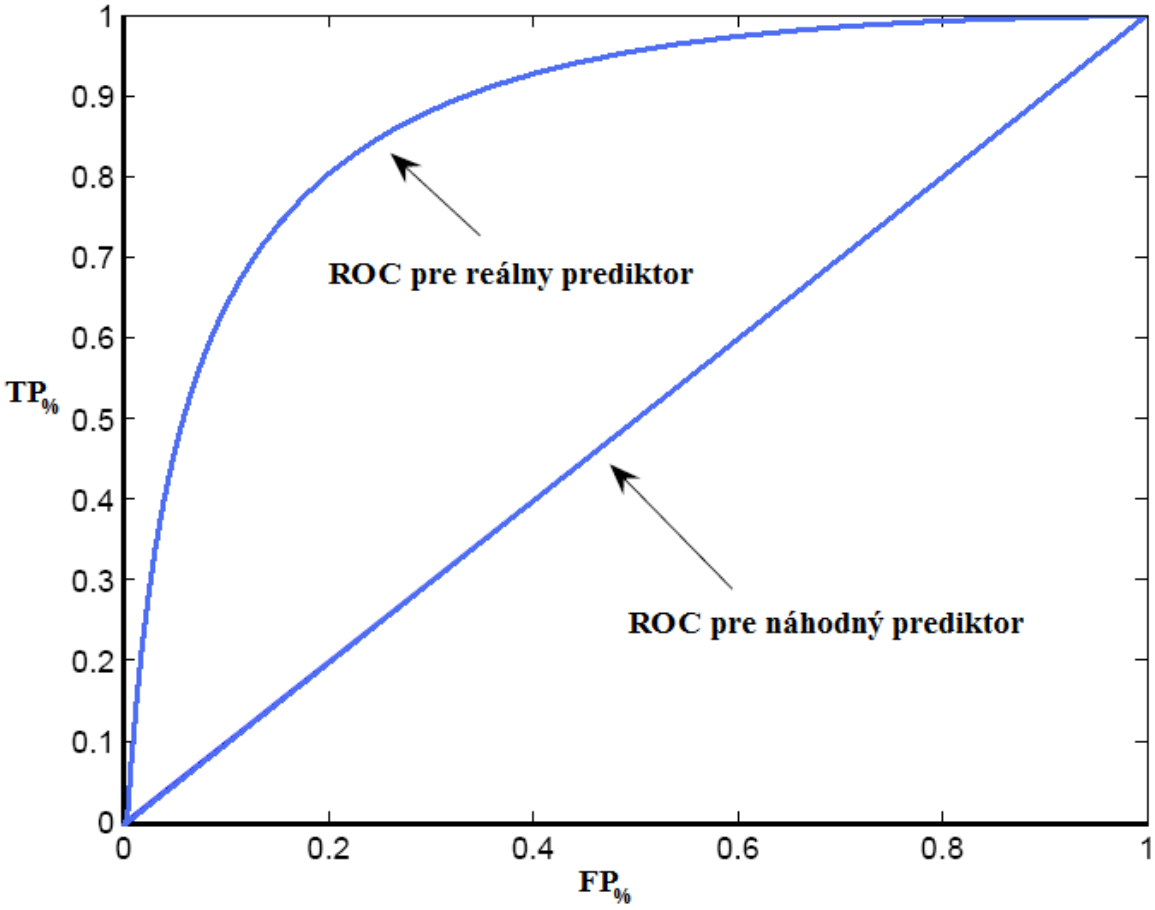
Krivka ROC (Receiver operating characteristic curve) bola prevzatá pre evalváciu modelov z oblasti rádiotechniky pomerne nedávno. Táto krivka dáva do súvislosti podiel TP s podielom FP

$$\mathrm{TP}_{\%} = \frac{TP}{TP + FN}$$

$$\mathrm{FP}_{\%} = \frac{FP}{FP + TN}$$

Tieto charakteristiky sú známe: $TP_{\%}$ = Sensitivita, $FP_{\%}$ = 1 – Špecificita. Používa sa teda aj kritérium $TP_{\%}*(1-FP_{\%})$.

ROC krivku je možné použiť pri modeloch, ktoré klasifikáciu dopĺňajú váhou resp. pravdepodobnosťou. Krivku vytvoríme tak, že budeme meniť prah pri ktorom bude výsledok klasifikácie interpretovaný ako „+“. Bodu (0,0)(žiadna predikcia triedy “+”) zodpovedá prahu 1. Bodu (1,1) zodpovedá prah 0. Zmenou prahu možno simulovať chovanie modelu v prípade zmeny pomeru medzi počtami príkladov oboch tried a aj zmeny cien za chybu v klasifikácií. ROC krivka teda dáva obraz o chovaní klasifikátoru bez ohľadu na rozdelenie tried a cenu chyby.



Obrázok 7

Je žiaduce sa v grafe pohybovať vľavo hore, teda v blízkosti bodu [0,1] , ktorý zodpovedá bezchybnej klasifikácií. [13]

2.7.8 AUC - plocha pod ROC krivkou

AUC (Area Under the ROC Curve) čiže plocha pod ROC krivkou je najbežnejší kvantitatívny index popisujúci ROC krivku. Je užitočné napríklad pri porovnávaní dvoch ROC kriviek, keď celú ROC krivku zredukuje do jednej skalárnej veličiny. Keďže je AUC časťou jednotkového štvorca, jeho hodnota bude vždy medzi 0 a 1. Z grafu (Obr. 4) vidíme že plocha pod ROC krivkov nikdy nebude nadobúdať menšie hodnoty ako 0,5 čo je hodnota dosiahnutá pre náhodný prediktor. Preto bude AUC nadobúdať iba hodnoty z intervalu (0,5;1,0)

Nasledujúca tabuľka ukazuje ohodnotenie presnosti testu:

veľkosť AUC	hodnotenie
0,9 – 1,0	výborné
0,8 – 0,9	veľmi dobré
0,7 – 0,8	dobré
0,6 – 0,7	dostatočné

0,5 – 0,6	nedostatočné

Tabuľka 2

Jednou z možných interpretácií plochy AUC je, že plocha AUC sa rovná pravdepodobnosti, že náhodne vybraný jedinec z triedy negatívnych dopadol v testovaní horšie ako náhodne vybraný jedinec z triedy pozitívnych.[13]

2.7.9 Gini koeficient

AUC je priamo spätý s Gini koeficientom a je ho možné z AUC priamo odvodiť:

$$\text{Gini} + 1 = 2 * \text{AUC}$$

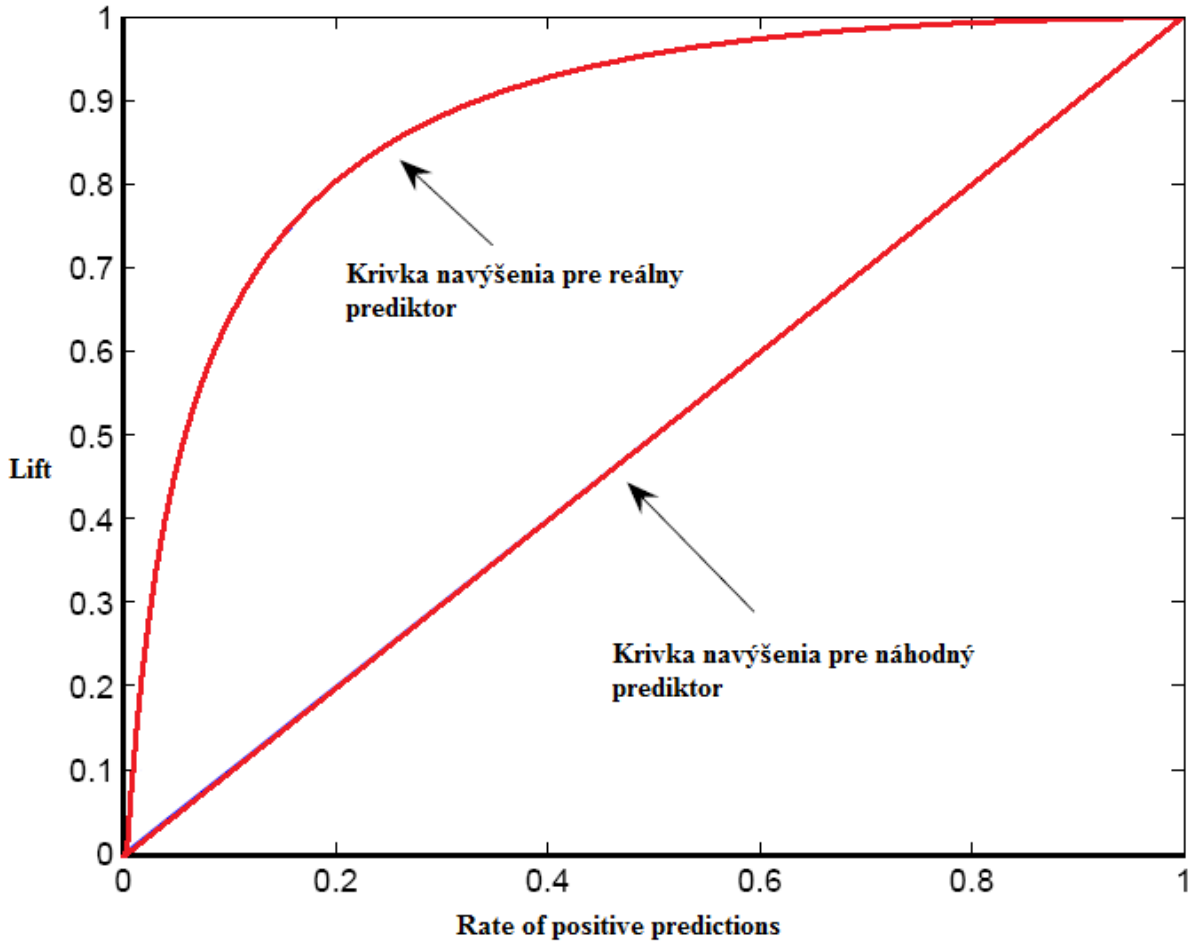
Gini koeficient predstavuje číselnú charakteristiku diverzifikácie resp. mieru diverzifikačnej schopnosti klasifikačného modelu. [13]

2.7.10 Krivka navýšenia

Krivka navýšenia, nazývaná aj „Lift Chart“ je podobná ROC krivke. Krivka navýšenia dáva do súvislosti senzitivitu klasifikátora s celkovým počtom prvkov datasete. To je možné iba pri modeloch, ktoré neposkytujú výlučne binárne výsledky klasifikácie, ale poskytujú aj informáciu o pravdepodobnom zaradení do triedy, ktorá je vyjadrená numerickou hodnotou.
Tak ako aj ROC krivka aj krivka navýšenia má na svojom definičnom obore všetky klasifikátory. Tieto klasifikátory zoradíme podľa pravdepodobnosti zaradenia (váhy) klasifikátora. O každom prípade máme informáciu do ktorej tiedy má byť zaradený. Pre každý klasifikátor vypočítame hodnoty podľa rovnice:

$$\text{x} = \frac{TP + FP}{P + N}, y = TP$$

Tieto hodnoty potom vynášame na graf a tak získame krivku navýšenia. Príklad takéhoto grafu je na obrázku 8.



Obrázok 8

Model je tým lepší, čím leží krivka navýšenia bližšie k diagonále, ktorá reprezentuje náhodný výber. Takisto ako aj ROC krivku, aj krivku navýšenia je možné číselne vyjadriť pomocou AUC, vypočítaním plochy pod krivkou navýšenia.

3 Dataset pre experimenty

Dataset pre experimenty obsahuje dáta génových expresií z 89 vzoriek nádorov rakoviny prsníka. Každá vzorka obsahuje 12625 údajov o expresií génov ktoré boli sledované. Tieto vzorky predstavujú heterogénnu populáciu a boli vybrané na základe klinických parametrov ktoré sa na pacientoch s týmito vzorkami nádorov prejavili.

Vekové rozloženie skúmaných pacientov:

Vek	Počet (%)
<40	27 (30%)
41–50	26 (29%)
51–60	19 (21%)
>60	17 (19%)

Tabuľka 3

3.1 Metódy použité pre získanie datasetu

- Vzorky nádorov boli získané biopsiou v Koo Foundation Sun Yat-Sen Cancer Centre Taipei v priebehu rokov 1991 až 2001.
- RNA zo vzoriek bola extrahovaná použitím Qiagen RNEasy kitov.
- DNA čip použitý pre získanie génových expresií bol triedy Affymetrix, konkrétne typ U95Av2 určený pre skúmanie ľudskej DNA. Tento čip obsahuje 12625 spotov.
- Čipy boli zosnímané pomocou Affymetrix GeneArray skeneru. Expresie génov boli vypočítané pomocou Affymetrix microarray suite verzie 5.0 a potom konvertované $\log_2$ škálou. [3]

3.2 Klinické charakteristiky vzoriek

Na 37 vzorkách bola skúmaná existencia metastáz na odobratých lymfatických uzlinách. Podľa toho sú rozdelené na triedu bez výskytu týchto metastáz (19 vzoriek) a s ich výskytom (18 vzoriek).
Na zvyšných 52 vzorkách bol skúmaný opätovný výskyt rakoviny v priebehu nasledujúcich troch rokov od odobratia vzorky. Podľa tohto kritéria sú vzorky rozdelená na triedu bez opätovného výskytu (34 vzoriek) a s výskytom (18 vzoriek).

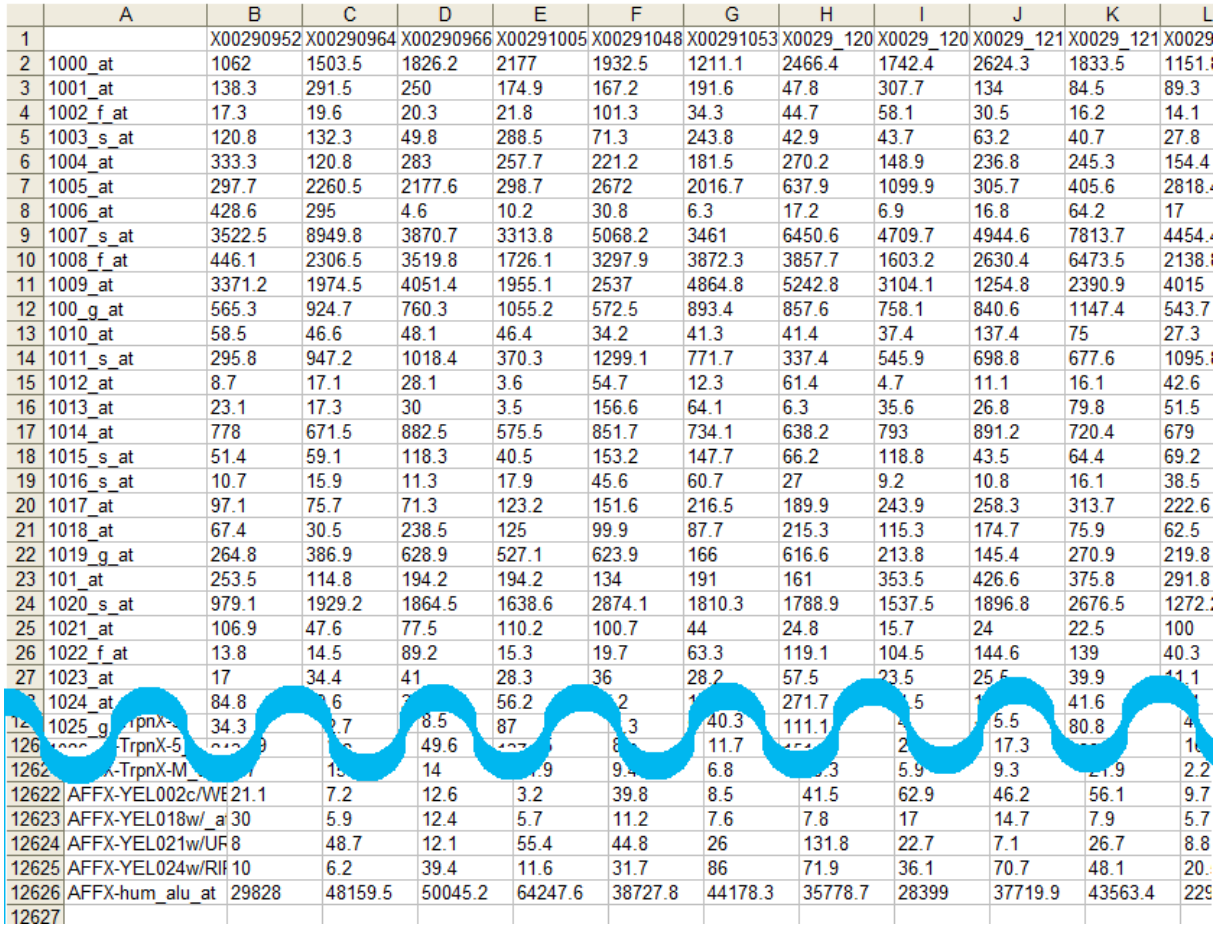
Skúmané klinické prejavy u pacientov:

	Nevyskytujúce sa metastázy na lymfatických uzlinách	Vyskytujúce sa metastázy na lymfatických uzlinách	Bez opätovného výskytu rakoviny do troch rokov.	S opätovným výskytom rakoviny do troch rokov.
1. skupina (37 pacientov)	19	18	nedefinované	nedefinované
2. skupina (52 pacientov)	nedefinované	nedefinované	34	18

Tabuľka 4

3.3 Popis súborov datasetu

Dataset je uložený v dvoch súboroch *microarray.csv* a *clinical.csv*. Oba sú vo formáte CSV (Comma-separated values), čiže uložené dáta sú oddelené čiarkami vďaka čomu sú spracovateľné na rôznych platformách. Súbor *microarray.csv* obsahuje samotné hodnoty expresií génov pre všetkých 89 pacientov označených v stĺpcoch. V hlavičkách riadkov sú označenia 12625 skúmaných génov (Obr. 5).



	A	B	C	D	E	F	G	H	I	J	K	L
1		X00290952	X00290964	X00290966	X00291005	X00291048	X00291053	X0029_120	X0029_120	X0029_121	X0029_121	X0029
2	1000_at	1062	1503.5	1826.2	2177	1932.5	1211.1	2466.4	1742.4	2624.3	1833.5	1151.
3	1001_at	138.3	291.5	250	174.9	167.2	191.6	47.8	307.7	134	84.5	89.3
4	1002_f_at	17.3	19.6	20.3	21.8	101.3	34.3	44.7	58.1	30.5	16.2	14.1
5	1003_s_at	120.8	132.3	49.8	288.5	71.3	243.8	42.9	43.7	63.2	40.7	27.8
6	1004_at	333.3	120.8	283	257.7	221.2	181.5	270.2	148.9	236.8	245.3	154.4
7	1005_at	297.7	2260.5	2177.6	298.7	2672	2016.7	637.9	1099.9	305.7	405.6	2818.
8	1006_at	428.6	295	4.6	10.2	30.8	6.3	17.2	6.9	16.8	64.2	17
9	1007_s_at	3522.5	8949.8	3870.7	3313.8	5068.2	3461	6450.6	4709.7	4944.6	7813.7	4454.
10	1008_f_at	446.1	2306.5	3519.8	1726.1	3297.9	3872.3	3857.7	1603.2	2630.4	6473.5	2138.
11	1009_at	3371.2	1974.5	4051.4	1955.1	2537	4864.8	5242.8	3104.1	1254.8	2390.9	4015
12	100_g_at	565.3	924.7	760.3	1055.2	572.5	893.4	857.6	758.1	840.6	1147.4	543.7
13	1010_at	58.5	46.6	48.1	46.4	34.2	41.3	41.4	37.4	137.4	75	27.3
14	1011_s_at	295.8	947.2	1018.4	370.3	1299.1	771.7	337.4	545.9	698.8	677.6	1095.
15	1012_at	8.7	17.1	28.1	3.6	54.7	12.3	61.4	4.7	11.1	16.1	42.6
16	1013_at	23.1	17.3	30	3.5	156.6	64.1	6.3	35.6	26.8	79.8	51.5
17	1014_at	778	671.5	882.5	575.5	851.7	734.1	638.2	793	891.2	720.4	679
18	1015_s_at	51.4	59.1	118.3	40.5	153.2	147.7	66.2	118.8	43.5	64.4	69.2
19	1016_s_at	10.7	15.9	11.3	17.9	45.6	60.7	27	9.2	10.8	16.1	38.5
20	1017_at	97.1	75.7	71.3	123.2	151.6	216.5	189.9	243.9	258.3	313.7	222.6
21	1018_at	67.4	30.5	238.5	125	99.9	87.7	215.3	115.3	174.7	75.9	62.5
22	1019_g_at	264.8	386.9	628.9	527.1	623.9	166	616.6	213.8	145.4	270.9	219.8
23	101_at	253.5	114.8	194.2	194.2	134	191	161	353.5	426.6	375.8	291.8
24	1020_s_at	979.1	1929.2	1864.5	1638.6	2874.1	1810.3	1788.9	1537.5	1896.8	2676.5	1272.
25	1021_at	106.9	47.6	77.5	110.2	100.7	44	24.8	15.7	24	22.5	100
26	1022_f_at	13.8	14.5	89.2	15.3	19.7	63.3	119.1	104.5	144.6	139	40.3
27	1023_at	17	34.4	41	28.3	36	28.2	57.5	23.5	25.6	39.9	11.1
	1024_at	84.8	6		56.2	2		271.7	.5		41.6	
	1025_g_	34.3	.7	8.5	87	3	40.3	111.1		5.5	80.8	
126		-TrpnX-5		49.6			11.7		2	17.3		
1262		X-TrpnX-M		14	7.9	9.4	6.8	.3	5.9	9.3	21.9	2.2
12622	AFFX-YEL002c/WE	21.1	7.2	12.6	3.2	39.8	8.5	41.5	62.9	46.2	56.1	9.7
12623	AFFX-YEL018w/_a	30	5.9	12.4	5.7	11.2	7.6	7.8	17	14.7	7.9	5.7
12624	AFFX-YEL021w/UF	8	48.7	12.1	55.4	44.8	26	131.8	22.7	7.1	26.7	8.8
12625	AFFX-YEL024w/RI	10	6.2	39.4	11.6	31.7	86	71.9	36.1	70.7	48.1	20.
12626	AFFX-hum_alu_at	29828	48159.5	50045.2	64247.6	38727.8	44178.3	35778.7	28399	37719.9	43563.4	229
12627												

Obrázok 9

Súbor *clinical.csv* obsahuje klinické informácie o pacientoch. Pre našu analýzu sú najdôležitejšie prvé dva stĺpce. Prvý stĺpec udáva či sa u pacienta vyskytli (označenie 1) , alebo nevyskytli (označenie 0) metastázy v lymfatických uzlinách. Druhý stĺpec udáva, či sa u pacienta opätovne vyskytla (označenie 1), alebo nevyskytla (označenie 0) rakovina do troch rokov od odoberania vzorky. Klinické vlastnosti označené v *i*-tom riadku prináležia pacientovi v *i*-tom stĺpci v súbore *microarray.csv*.

4 Realizácia systému využívajúceho Random Forests ku kategorizácii dát

Systém som realizoval v jazyku R, pomocou balíčka „randomForest“. Tento balíček je vytvorený podľa pôvodnej implementácie v jazyku Fortran priamo od autora metódy Lea Breimana.

4.1 Jazyk R

R [14] je jazyk a prostredie pre štatistické výpočty a tvorbu grafov. Je to voľne šíriteľný nástroj, ktorý vychádza z komerčného jazyka S. Medzi jazykmi je niekoľko dôležitých rozdielov, ale väčšina kódu pre jazyk S je bez modifikácií spustiteľná v prostredí jazyka R.
Tento jazyk bol primárne navrhnutý pre rôzne štatistické techniky, ako lineárne a nelineárne modelovanie, klasická štatistika, analýza časových radov, klasifikáciu, zhlukovanie a množstvo

ďalších. Zároveň podporuje aj rôzne grafické techniky, najmä pre tvorbu grafov. Zdrojový kód jazyka R je otvorený, vďaka čomu je rozšíriteľný. Veľkou výhodou jazyka je tiež že dokáže veľmi jednoducho generovať grafy s kvalitou vhodnou aj pre publikácie.
R je dostupný vo forme zdrojového kódu pod podmienkami GNU licencie a je kompilovateľný a spustiteľný na širokom spektre UNIX platforiem, Windows a MacOS.
Jazyk a prostredie R poskytuje:

- Efektívne spracovanie a ukladanie dát
- Skupinu operátorov pre výpočty nad poľami, resp. maticami dát
- Veľký ucelený súbor nástrojov pre analýzu dát
- Grafické prostriedky pre analýzu dát a ich zobrazenie
- Dobre vyvinutý, jednoduchý a efektívny programovací jazyk ktorý zahŕňa podmienené funkcie, cykly a užívateľom definované rekurzívne funkcie a prostriedky pre vstup a výstup dát

Prostredie je rozšíriteľné buď priamo v jazyku R, alebo pre výpočetne intenzívnejšie úlohy je možné nalinkovanie z kódu v jazykoch C, C++ a Fortran. Existuje dokonca možnosť napísať kód v jazyku C pre priamu manipuláciu s R objektmi.
Do prostredia R je možné importovať rozšírenia v podobe balíčkov ktoré rozširujú jeho výpočtové schopnosti a pridávajú mu vlastnosti ktoré užívateľ potrebuje. Široké spektrum balíčkov je prístupných na internete v archíve CRAN (Comprehensive R Archive Network). Každý z tu dostupných balíčkov je navyše podľa štandardu R dobre zdokumentovaný pre jednoduchšie použitie.

4.2 Balíček randomForest

Balíček s názvom randomForest[15] obsahuje priamo pôvodnú implementáciu metódy Random Forests od jej autora Lea Breimana. Vďaka veľkej flexibilite jazyka R bolo možné pôvodnú implementáciu ktorá je napísaná v jazyku Fortran pripojiť do balíčka prostredia R.
Balíček je primárne určený pre klasifikáciu a regresiu metódou Random Forests. Ďalej poskytuje široké spektrum nástrojov ktorými je možne analýzu dát špecifikovať a prispôsobiť potrebám používateľa. Navyše obsahuje aj funkcie pre tvorbu grafov špeciálne určených pre reprezentáciu výsledkov analýzy metódou Random Forests.

4.2.1 Funkcia randomForest

Je to hlavná funkcia balíčka. Implementuje klasifikáciu a regresiu metódy Random Forests. Môže byť použitá aj v móde učenia bez učiteľa. V tomto prípade poskytuje informácie o podobnosti prvkov.
Funkcia má viac ako 20 rôznych argumentov. Nie je potrebné zadávať všetky parametre. Pre väčšinu argumentov existujú predvolebné hodnoty ktoré sa použijú ak používateľ argument nezadá.

Dôležité argumenty funkcie **randomForest**:

- **data** – je voľitelný argument obsahujúci premenné modelu. Predvolene sa berú premenné z prostredia v ktorom je funkcia spúštaná.
- **subset** – je index vektorov indikujúci, ktoré riadky z datasetu sa majú použiť.

- **na.action** – určuje aká funkcia má byť použitá v prípade že dataset obsahuje chýbajúce údaje.
- **x** alebo **formula** – matica prediktorov
- **y** – zaradenie prvkov z **x** do triedy.
- **ntree –** nastavenie počtu generovaných stromov. Nemala by to byť príliš malá hodnota, aby bolo zaručené, že každý riadok vstupu bude predikovaný aspoň niekoľko krát. Táto hodnota je prednastavená na 500.
- **mtry –** nastavenie počtu premenných ktoré budú náhodne vybrané pri delení uzlu stromu. Predvolené hodnoty tohto argumentu sú rozdielne pre klasifikáciu (odmocnina počtu premenných v x) a pre regresiu (p/3).
- **replace –** určuje či sa pri delení stromu použije bootstrapový výber. Hodnota je prednastavená na TRUE.

Výstupom funkcie randomForest je objekt triedy randomForest. Tento objekt je vlastne klasifikátor (v prípade klasifikácie) vytvorený na základe trénovacích dát. Obsahuje zoznam argumentov s akými bolo trénovanie vykonávané, čiže s akými parametrami bola volaná funkcia randomForest. Okrem toho obsahuje aj hodnoty, ktoré popisujú vlastnosti vytvoreného klasifikátora.

Dôležité hodnoty objektu triedy **randomForest**:

- **type** – môže nadobúdať hodnoty *regression, classification,* alebo *unsupervised***.** Informuje v akom móde prebiehala funkcia randomForest. Hodnota *unsupervised* značí že prebiehala v móde bez učiteľa.
- **predicted** – hodnoty predikované na základe OOB vzoriek zo vstupných dát.
- **importance** – matica významov premenných pre klasifikačné triedy. Navyše predposledných stĺpec obsahuje význam premenných vypočítaný nad všetkými klasifikačnými triedami. Posledný stĺpec obsahuje Gini index.
- **forest** – zoznam ktorý obsahuje celý vygenerovaný klasifikačný les.
- **err.rate** – vektor chybovosti predikcie na vstupných dátach.
- **votes** – matica s jediným riadkom pre každý prvok zo vstupných dát a stĺpcom pre každú **triedu.** Informuje o hlasovaniach pre triedy.

4.2.2 Funkcia predict.randomForest

Je to funkcia pre klasifikáciu testovaných dát pomocou metódy Random Forest.
Dôležité argumenty funkcie **predict.randomForest**:

- **object** – objekt triedy randomForest vytvorený funkciou randomForest.
- **newdata** – matica obsahujúca testovacie dáta ktorých triedu chceme predikovať.
- **type** – určuje typ výstupných dát. Môže nadobúdať hodnoty *response* , *prob.* , alebo *votes***.**
- **proximity** – určuje či majú byť počítané proximity (príbuznosti objektov).

- **nodes** – určuje či majú byť súčasťou výstupu aj hlasovania terminálnych ulov stromov.

Výstupy funkcie sa líšia podľa nastavenia jej argumentov. Ak bol vstupný objekt typu *regresion*, výstupom bude vektor predikovaných hodnôt. Ak bol vstupný objekt typu *classification* výstup závisí na hodnote argumentu *type*:

- pre hodnotu *response* je výstupom matica zaradenia testovacích objektov do tried.
- pre hodnotu *prob*. je výstupom matica pravdepodobného zaradenia objektov do tried.
- pre hodnotu *votes* je výstupom matica hlasov pre triedy.

V prípade že v argumentoch predikcie je nastavená na TRUE hodnota *proximity*, resp. *nodes* sú súčasťou výstupu aj vypočítané proximity, resp. hlasovania terminálnych uzlov stromov.

4.2.3 Funkcia getTree

Je funkcia, ktorá extrahuje konkrétny strom s celou jeho štruktúrou z klasifikátora objektu triedy randomForest.
Argumentmi funkcie sú:

- **rfobj –** objekt triedy randomForest.
- **k –** index stromu ktorý chceme získať.
- **labelVar –** určuje, či má byť výstupom funkcie prehľadná matica s označenými uzlami.

Výstupom funkcie getTree bude v prípade hodnoty TRUE argumentu labelVar matica so šiestimi stĺpcami a s počtom riadkov podľa počtu uzlov v strome. Stĺpce matice predstavujú:

- **left daughter –** číslo riadku s ľavým následovníkom uzlu. Hodnota 0 ak sa jedná o uzol terminálny.
- **right daughter -** číslo riadku s pravým následovníkom uzlu. Hodnota 0 ak sa jedná o uzol terminálny.
- **split var –** premenná ktorá bola použitá pre delenie uzlu.
- **split point –** pozícia najlepšieho rozdelenia.
- **status –** hodnota -1 ak sa jedná o uzol terminálny, inak hodnota 1.
- **prediction –** predikcia uzla. Nadobúda hodnotu 0 ak sa jedná o uzol neterminálny.

Funkcia **importance –** funkcia ktorá extrahuje významnosť premenných z objektu triedy randomForest.
Argumentmi funkcie sú:

- **x –** objekt triedy randomForest.
- **type –** Špecifikuje typ výpočtu významnosti premenných môže nadobúdať hodnotu 1, alebo 2 (1 pre zníženú presnosť, 2 pre zníženú exaktnosť uzla)

- **labelVar** – určuje, či má byť výstupom funkcie prehľadná matica s označenými uzlami.
- **class** – špecifikuje, pre ktorú triedu sa majú významnosti počítať.
- **scale** – určuje, či sa majú hodnoty vydeliť hodnotou „standard error“

Výstupom funkcie je vektor významnosti pre každý prediktor premennej.

4.3 Balíček e1071

Tento balíček bol vytvorený na katedre Štatistiky Technickej Univerzity vo Viedni. Použil som ho pre SVM klasifikáciu. Okrem toho poskytuje aj ďalšie funkcie pre štatistickú analýzu dát, ako napríklad fuzzy zhlukovanie, Bayesovská klasifikácia a ďalšie. Priblížime si funkcie SVM klasifikátora v tomto balíčku.

4.3.1 Funkcia svm

Táto funkcia slúži na natrénovanie klasifikátora. Okrem klasifikácie je schopná aj regresie.
Základné parametre funkcie **svm** sú:

- formula – symbolický popis modelu
- data – vstupné dáta ktoré obsahujú premennú podľa ktorej chceme trénovať
- x – matica premenných podľa ktorých v prípade klasifikácie trénujeme klasifikátor
- y – zaradenie prvkov z **x** do triedy. Pre každý riadok jeden. Pre klasifikáciu musí byť typu factor.
- type – typ natrénovania, klasifikácia, alebo regresia
- kernel – typ kernelovej funkcie ktorá má byť použitá. K dispozícií sú typy
    - linear – pre lineárnu kernelovú funkciu
    - polynomial – pre polynomiálnu kernelovú funkciu
    - radial basis – pre radiálnu kernelovú funkciu
    - sigmoid – pre sigmoidálnu kernelovú funkciu
- subset – parameter pre určenie podmnožiny použitej pri trénovaní. Zvyšok je možné použiť napríklad pre zistenie kvality klasifikátora.
- gama – gama parameter potrebný pre každú kernelovú funkciu okrem linearnej. Prednastavený je na hodnotu 1/počet premenných datasetu
- cost – množstvo povolených nesprávne klasifikovaných prvkov, prednastavené na 1

Takisto ako funkcia v balíčku randomForest, aj v e1071 je funkcia prediction (prediction.svm) ktorá klasifikuje dáta na základe objektu typu svm. Vstupné parametre sú pre nu podobné ako v randomForests.

4.3.2 Funkcia tune

4.4 Balíček rpart

Tento balíček poskytuje nástreje pre vytváranie jednoduchých klasifikačných stromov. Na rozdiel od od podobného balíčka tree, tento poskytuje navyše aj možnosti pre rôzne metódy pre vytváranie klasifikačných stromov.

4.4.1 Funkcia rpart

Funkcia pre vytváranie klasifikačného stromu. Vytvára klasifikátor do objektu typu rpart. Základné parametre funkcie rpart, sú podobné ako v predchádzajúcich balíčkoch. Takisto je to formula, data, x, y, subset. Výnimočný je parameter method, ktorý nastavuje metódu ktorá sa má použiť pre delenie uzla pri vytváraní stromu. Môže mať hodnoty:

- anova
- poisson
- class
- exp

Aj v tomto balíčku je funkcia predict (predict.rpar). Veľmi praktickou je aj funkcia prune.rpart, ktorá umožňuje orezávanie stromu.

4.5 Balíček nnet

Je to balíček pre vytváranie dopredných neurónových sietí s jednou skrytou vrstvou.

4.5.1 Funkcia nnet

Je to Funkcia pre natrénovanie neurónovej siete s jedno skrytou vrstvou. Takisto má základné parametre ako trénovacie funkcie v predchádzajúcich balíčkoch. Okrem toho sú to:

- weights – nastavenie váh pre vstupnú vrstvu. Predvolená hodnota je 1.
- size – nastavuje počet neurónov v skrytej vrstve.
- MaxNWts – parameter pre nastavenie maximálneho povoleného počtu váh.
- maxit – maximálny počet iterácií.

Aj v tomto balíčku je funkcia predict (predict.nnet) pre klasifikáciu podľa natrénovanej neurónovej siete.

4.6 Balíček ROCR

Balíček ROCR[16] slúži na analýzu a porovnávanie klasifikátorov. Poskytuje až 25 rôznych metód pre hodnotenie kvality klasifikátorov, vrátane najpoužívanejších ako celková správnosť, celková chyba, špecificita, senzitivita. Jeho najväčšou prednosťou je že dokáže tieto hodnoty efektívne zobrazovať v grafoch súvislosti, napr. ROC krivka. Balíček pracuje s testovacími dátami generovanými krížovou validáciou, bootstrapovým výberom, alebo inou testovacou metódou. Celý proces prebieha v iba troch funkciách.

4.6.1 Funkcia prediction

Každé hodnotenie klasifikátora v ROCK balíčku začína v tejto funkcií, v ktorej sa vytvorí objekt triedy *prediction*. Táto funkcia transformuje vstupné dáta na štandardizovaný formát ktorý môže byť ďalej spracovaný.
Argumentmi funkcie sú:

- predictions – vektor, matica, zoznam, alebo data frame obsahujúci predikcie
- labels - vektor, matica, zoznam, alebo data frame obsahujúci skutočné zaradenie objektov do triedy. Musí mať rovnaký počet dimenzií ako *predictions*.
- label.ordering – nastavenie druhu tried do ktorých sú objekty zaradené

Výstupom funkcie je objekt triedy *prediction* ktorý je vstupom pre funkciu *performance*.

4.6.2 Funkcia performance

Vykonáva ohodnotenie klasifikátora z objektu triedy *prediction*. Dokáže vykonávať až 25 najčastejšie používaných charakteristík klasifikátorov. Taktiež má rozhranie pre individuálnu tvorbu užívateľských prepočtov.
Argumentmi funkcie sú:

- prediction.obj – objekt triedy *prediction*.
- measure – charakteristika klasifikátora ktorá sa použije pre hodnotenie.
- x.measure – druhá charakteristika, ktorá sa dá do súvislosti s *measure*.
- Ďalšie argumenty môžu byt použité pre definovanie vlastných charakteristík.

Príklad použitia:
Ak chceme získať napríklad ROC krivku ktorá zobrazuje vzťah medzi TP a FP, zvolíme argumenty nasledovne: measure = “tpr“ , x.measure = “fpr“ . Označenia všetkých 25 charakteristík je možné nájsť v manuáli k balíčku.

Výstupom funkcie je objekt triedy *performance* ktorý je použiteľný pre generovanie grafických výstupov funkciami pre generovanie grafov.

5 Testovanie systému na datasete

Testy boli vykonávané v prostredí jazyka R. Pretože sa jedná o pamäťovo aj výpočtovo vysoko náročné operácie, použil som pre testovanie školou poskytnutý server athena2, s konfiguráciou:

- Supermicro H8DME,
- 2xDual Core AMD Opteron 2220,
- 8/16 GB RAM, 300 GB HDD
- CentOS 5.4 64bit Linux

Tento výkonný server poskytuje dostatok operačnej pamäte pre operovanie nad tak vysokodimenzionálnym datasetom aký je testovaný v tejto práci. Poskytuje až 4 výpočtové jadrá. Avšak výpočtový nástroj R dokáže pracovať vždy len s jedným jadrom. Jediný paralelizovateľný

klasifikátor z tých ktoré sú použité v tejto práci je randomForests, ktorý dokáže pomocou balíčka foreach rodeľiť generovanie stromov do viacerých procesov. Na rýchlosť generovania to však malo minimálny vplyv.
Pre generovanie klasifikátorov boli použité balíčky popísané v predchádzajúcej kapitole.

5.1 Klasifikácia metódou Random Forests

5.1.1 Klasifikácia podľa metastáz na lymfatických uzlinách

Pre klasifikáciu podľa metastáz na lymfatických uzlinách je použitých prvých 37 vzoriek z datasetu. Ostatné nemajú tento údaj definovaný. Z týchto 37 vzoriek je prvých 19 negatívnych (označenie 0) a zvyšných 18 pozitívnych (označenie 1). Náležitosť do tried je teda rovnomerne rozdelená.

5.1.1.1 Klasifikácia s prednastavenými hodnotami

Najprv som pre trénovanie klasifikátora na skúmanom datasete použil funkciu randomForest so štandardami argumentmi.

Natrénovanie klasifikátora prebehlo s generovaním 500 stromov a veľkostou bootstrapového výberu označovaného mtry s hodnotou 112. Hodnota mtry zodpovedá $\sqrt{n}$, kde *n* predstavuje počet premenných ($\sqrt{12625} \approx 112$). Takto nastavená hodnota by sa mala podľa Breimana blížiť ideálnej hodnote pre klasifikáciu[17]. Keďže algoritmus Random Forests generuje vyhodnocovanie kvality klasifikácie už počas trénovania pomocou vzoriek, ktoré neboli zaradené do bootstrapového výberu, súčasťou výstupu funkcie randomForest nielen klasifikátor, ale aj OOB miera chyby.

Klasifikátor dosiahol nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **32.43%**
- Miera správnosti pre jednotlivé triedy je Acc_1= 0.6363, Acc_0 = 0.7333
- Matica zámen je v tabuľke (Tab. 5)

	Klasifikácia systémom	
Správne zaradenie	1	0
1	14	4
0	8	11

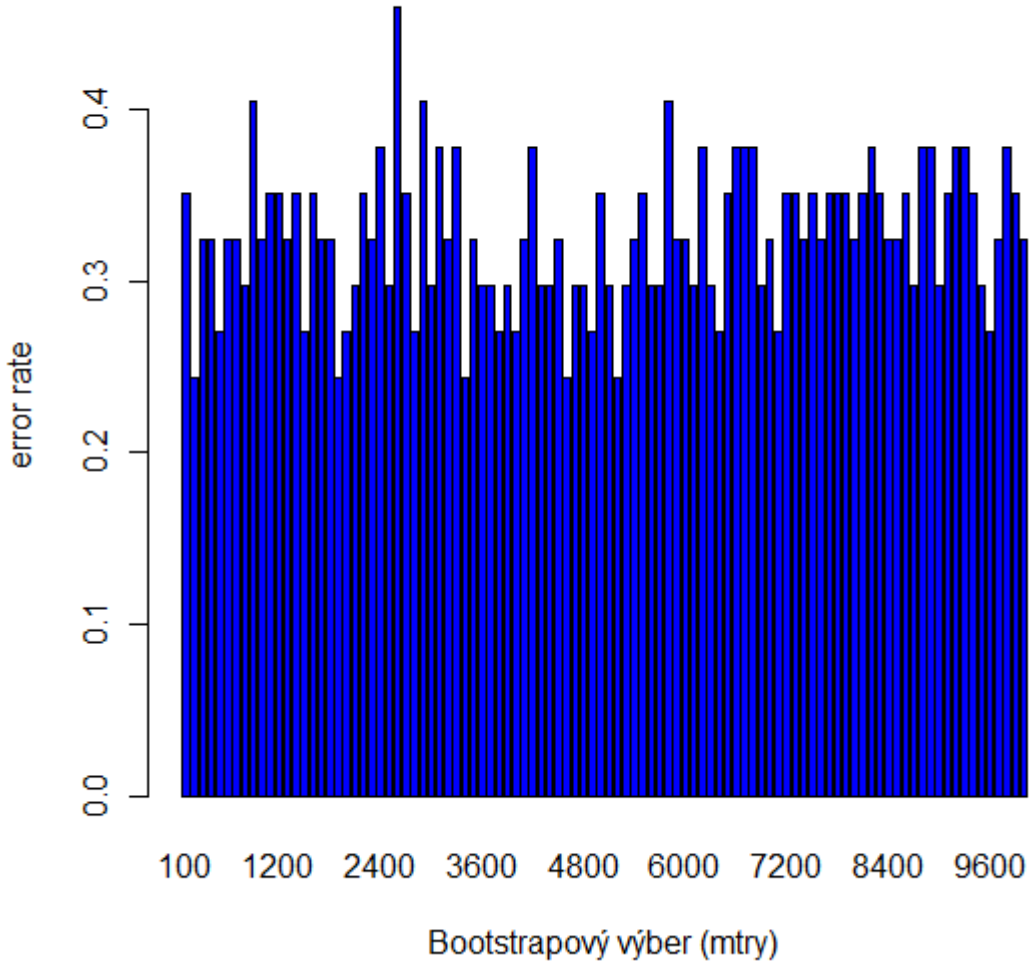
Tabuľka 5

Podľa tabuľky zámen bolo z negatívnych vzoriek 11 klasifikovaných správne a 8 nesprávne.
Z pozitívnych vzoriek bolo 14 klasifikovaných správne a 4 boli klasifikované nesprávne.

5.1.1.2 Vplyv veľkosti bootstrapového výberu

Podľa Breimana[1] je jediným nastaviteľným parametrom na ktorý je metóda Random Forests skutočne citlivá veľkosť bootstrapového výberu. Navrhuje aby sa ako hodnota mtry vybrala tá ktorá dosahuje najnižšiu mierou OOB chyby z trojice $\sqrt{n}$, $\frac{\sqrt{n}}{2}$, $2 * \sqrt{n}$.

Ja som sa rozhodol citlivosť metódy na hodnotu mtry preskúmať podrobnejšie. V cykle som 100 krát natrénoval klasifikátor s parametrom mrty v rozsahu 100 až 10 000. V každom kroku sa hodnota mtry zväčšuje o 100. Ostatné parametre som ponechal nezmenené. Výsledné miery chyby (error rate) pre každý z vygenerovaných klasifikátorov je v grafe(Obr. 10).

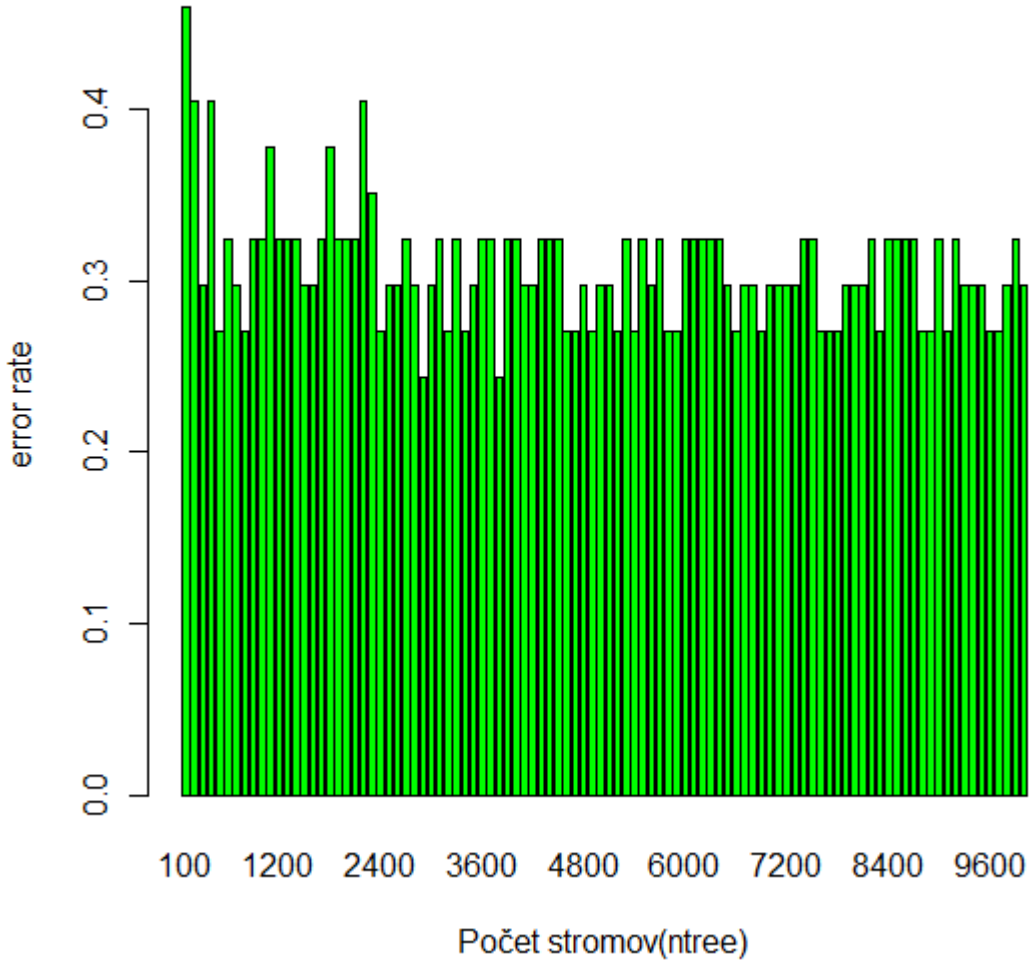


Obrázok 10

Z grafu je zrejmé, že v našom prípade sa miera chyby klasifikátorov pohybuje približne v rozmedzí 0,2 až 0,4. Zmena hodnoty mtry v rozmedzí 100 až 10 000 nemá na úspešnosť klasifikácie výraznejší vplyv.

5.1.1.3 Vplyv počtu stromov

Pre zistenie vplyvu počtu stromov na kvalitu klasifikátora som použil obdobný postup ako v prípade veľkosti bootstrapového výberu. Natrénoval som 100 krát klasifikátor s parametrom počtu geneovaných stromov (ntree) od hodnoty 100 až po hodnotu 10 000 s veľkosťou kroku 100. Výsledné miery chyby (error rate) pre každý z vygenerovaných klasifikátorov je v grafe (Obr. 11).



Obrázok 11

V grafe je možné pozorovať, že príliš malý počet generovaných stromov mal negatívny vplyv na kvalitu klasifikátora. So zvyšujúcim sa počtom generovaných stromov sa kvalita klasifikátorov zvyšuje. Miera chyby sa od počtu 2000 stromov ustaľuje na úrovni približne 0,3.

5.1.1.4 Optimálny klasifikátor

Na základe predchádzajúcich výsledkov som hľadal čo najoptimálnejší klasifikátor pre túto časť datasetu. Ako parametre funkcie randomForest som zvolil hodnoty mtry = 500 a ntree = 300. Pretože generovanie klasifikátora je spojené s určitou mierou náhodnosti je vhodné pre čo najlepší výsledok generovanie opakovať.

Vytvoril som jednoduchý skript ktorý v cykle 50 krát generuje klasifikátor so zadanými parametramy a hľadá ten s najnižšou mierou chyby. Nájdený klasifikátor bude použitý pri vyhodnocovaní výsledkov.

5.1.2 Klasifikácia podľa opätovného výskytu rakoviny

Pre klasifikáciu podľa opätovného výskytu rakoviny je použiteľných zvyšných 52 vzoriek z datasetu. Ostatné nemajú tento údaj definovaný. Z týchto 52 vzoriek je prvých 34 negatívnych (označenie 0) a zvyšných 18 pozitívnych (označenie 1). Náležitosť do tried nie je rovnomerne rozdelená.

5.1.2.1 Klasifikácia s prednastavenými hodnotami

Obdobne ako v prípade prvej časti datasetu som najprv vygeneroval klasifikátor s prednastavenými parametrami. Čiže mtry = 112 a ntree=500. Klasifikátor dosiahol nasledujúce výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **25%.**
- Miera správnosti pre jednotlivé triedy je Acc_1= 0.7777, Acc_0 = 0.7441
- Matica zámen je v tabuľke (Tab. 7)

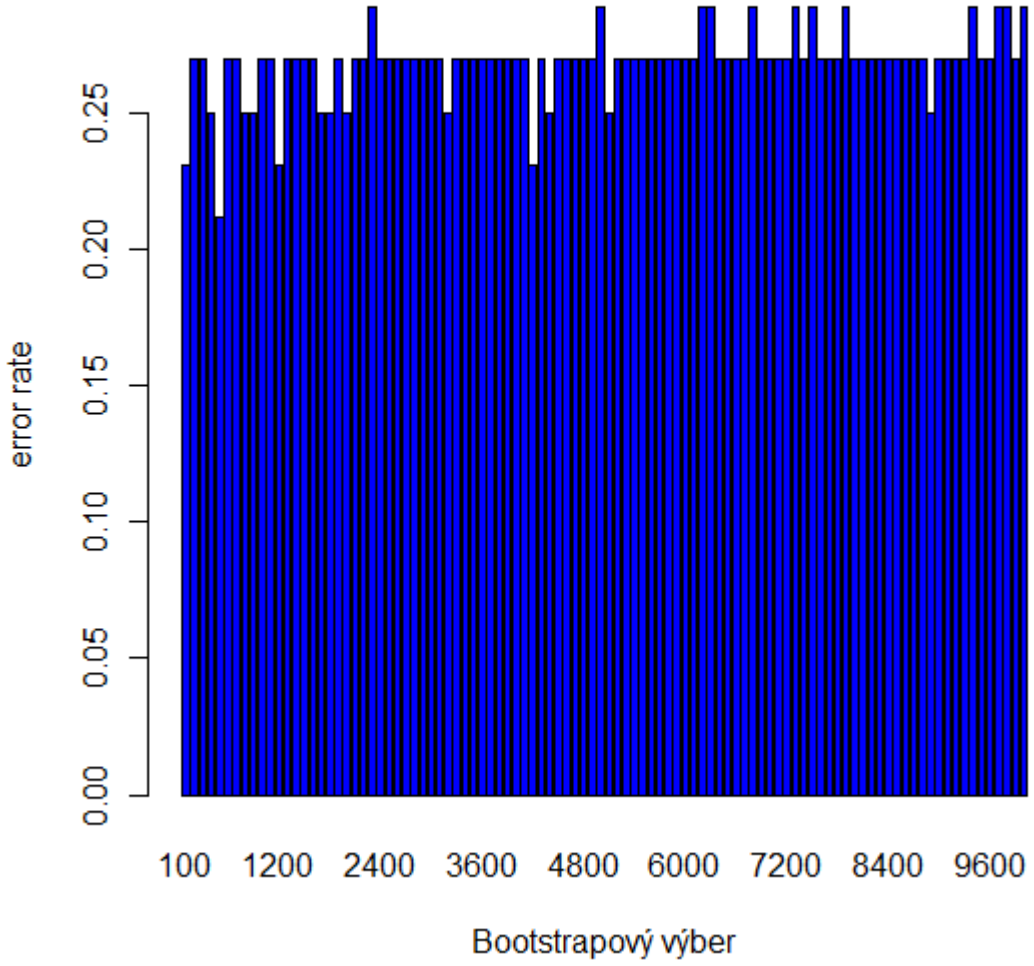
	Klasifikácia systémom	
Správne zaradenie	1	0
1	7	11
0	2	32

Tabuľka 6

Podľa tabuľky zámen dokáže klasifikátor výrazne lepšie zaradiť vzorku z triedy 0 ako z triedy 1. Vypovedá o tom aj hodnota Acc_0 = 0.9412, ktoá je výrazne vyššia ako Acc_1= 0.3889.To je pravdepodobne spôsobené výrazne väčšou trénovacou sadou pre triedu 0.

5.1.2.2 Vplyv veľkosti bootstrapového výberu

Pre zistenie vplyvu hodnoty mtry pre generovanie klasifikátora na tejto časti datasetu som použil rovnaký postup ako v prípade prvej časti. Čiže vygenerovanie 100 klasifikátorov s mtry v rozpätí 100 až 10 000.



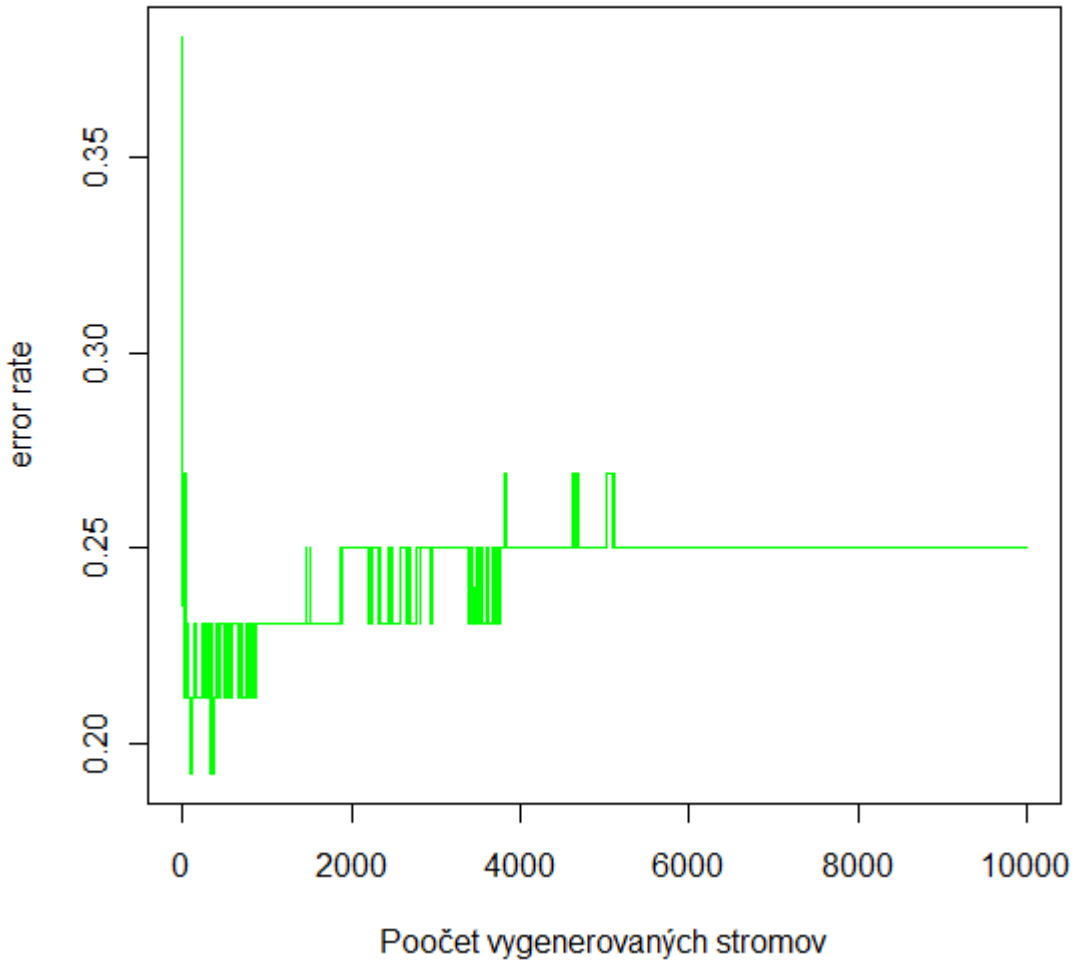
Obrázok 12

Z grafu (Obr. 12) vyplýva o niečo nižšia chybovosť pri nižších hodnotách mtry. V priemere sa však miera chyby pohybuje v rozmedzí 0,25 až 0,3.

5.1.2.3 Vplyv počtu stromov

Pre zistenie hodnoty počtu stromov pre klasifikátor s čo najnižšou mierou chyby som v tomto prípade použil iný postup ako v prvej časti datasetu. Použil som postup(odkaz), ktorý navrhuje autor metódy Breiman. Navrhuje vygenerovať klasifikátor s čo najväčším počtom stromov. Z priebehu vývoja chybovosti klasifikátora počas generovania stromov je možné zistiť, od akého počtu stromov sa už klasifikátor výraznejšie nezlepšuje. Tento postup je možný iba vďaka tomu, že metóda Random Forest odhaduje mieru chyby z OOB priamo počas trénovania klasifikátora.

Vygeneroval som klasifikátor s hodnotou ntree=10000. Priebeh zmeny miery chyby klasifikátora počas generovania stromov vidieť v grafe (Obr. 13)



Obrázok 13

Na priebehu grafu je zreteľne vidieť že miera chyby sa ustálila v okolí hodnoty 0,25 pri približne 2000 vygenerovaných stromoch.

5.1.2.4 Optimálny klasifikátor

Na základe predchádzajúcich výsledkov som hľadal čo najoptimálnejší klasifikátor pre túto časť datasetu. Ako parametre funkcie randomForest som zvolil hodnoty mtry = 200 a ntree = 2000. Pre nájdenie čo najlepšieho klasifikátora som použil rovnaký algoritmus ako pri prvej časti datasetu. Tento klasifikátor bude použitý pri vyhodnocovaní výsledkov.

5.2 Klasifikácia metódou SVM

Metóda SVM klasifikácie na rozdiel od Random Forests používa počas celej doby trénovania všetky vstupné dáta. Preto nie je možné ohodnocovať kvalitu klasifikátora už počas trénovania. Pre zistenie kvality klasifikácie som teda použil metódu leave-one-out, kde trénujeme klasifikátor bez jedného

vstupného objektu a následne na základe vytvoreného klasifikátora určíme triedu pre vynechaný objekt. Toto vykonáme pre každý objekt z trénovacej množiny. Celkovú chybovosť klasifikácie potom určíme podľa vzorca pre chybu klasifikácie.
V balíčku e1071 ktorý obsahuje nástroj pre SVM klasifikáciu existuje viacero možností nastavenia klasifikácie. Základným je nastavenie kernelovej funkcie. Natrénovania podľa typu kernelovej funkcie dosiahli nasledovné miery chybovosti:

- Lineárna funkcia: *Err* = 0,3513
- Polynomiálna funkcia *Err* = 0,687
- Radiálna funkcia *Err* = 0,675
- Sigmoidálna funkcia *Err* = 0,378

Okrem nastavenia kernelovej funkcie zostali ostatné parametre trénovania s prednastavenými hodnotami.

5.3 Klasifikácia metódou Rozhodovací strom

Pre natrénovanie klasifikátora som použil balíček rpart. Funkcia rpar automatický rozozná podľa vstupných údajov vhodný typ pestovania stromu. V tomto prípade typ class. Kvalitu netrénovaného klasifikátora určujeme podobne ako v prípade SVM pomocou leave-one-out.
Chybovosť klasifikácie dosiahla úroveň *Err* = 0,2092.

5.4 Klasifikácia metódou Neurónová sieť

Pre natrénovanie klasifikátora bol použitý balíček nnet. Pretože vzorky v datasete majú rozmer až 12625 údajov a každý údaj je vstupom pre každý neurón v skrytej vrstve, nie možné efektívne natrénovať neurónovú sieť s väčším počtom neurónov v skrytej vrstve. Zvýšenie počtu neurónov v skytej vrstve nemalo výraznejší vplyv na kvalitu klasifikácie.

6 Vyhodnotenie výsledkov

Výsledky klasifikácií jednotlivých metód som vyhodnotil podľa postupov popísaných v 2. Kapitole tejto práce. Pre každú klasifikačnú metódu je na oboch častiach datasetu vyhodnotená celková miera chyby, presnosť a správnosť zaradenia pre jednotlivé triedy. Vďaka tomu že všetky skúmané metódy klasifikácie poskytujú nielen absolútne zaradenie do triedy, ale aj pravdepodobnosť výsledného zaradenia, je pre každú metódu vygenerovaná ROC krivka a takisto aj plocha pod touto krivkou (AUC).

6.1 Vyhodnotenie metódy Random Forests

Vďaka tomu že metóda Random Forests generuje informácie o kvalite klasifikácie už počas trénovania, na základe prvkov ktoré neboli použité pre trénovanie konkrétneho stromu, poskytuje natrénovaný klasifikátor nezávislé odhady chyby samotného klasifikátora. Nie je preto nutná žiadna krížová validácia, ani iné metódy pre hodnotenie.

6.1.1 Vyhodnotenie výsledkov na prvej časti datasetu

Pre vyhodnotenie výsledkov klasifikácie prvej časti datasetu som použil klasifikátor, ktorého vygenerovanie je popísané v kapitole 5. Je to klasifikátor, ktorý dosahoval najlepšie výsledky na prvej časti datasetu. Dosiahol nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **24,32%**.
- Miera správnosti pre jednotlivé triedy je Acc_1= 0,7142, Acc_0 = 0,8125
- Presnosť = 0,8125 a Úplnosť = 0,6842
- Miera F = 0,7692
- Senzitivita = 0,6842, Špecificita = 0,8333
- Matica zámen je v tabuľke (Tab. 8)

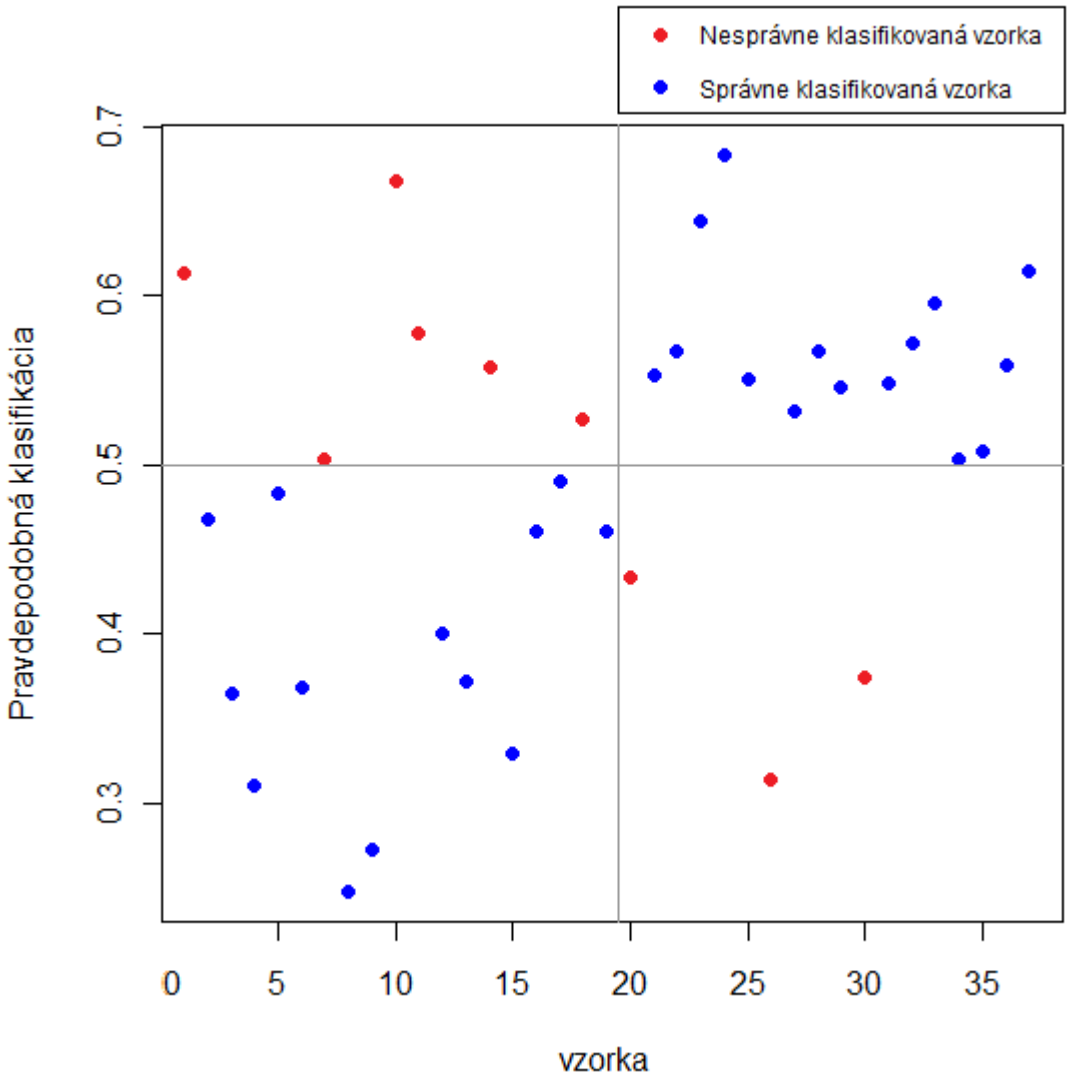
	Klasifikácia systémom	
Správne zaradenie	1	0
1	15	3
0	6	13

Tabuľka 7

Z tabuľky zámen ako aj z mier správnosti pre jednotlivé triedy vidíme, že o niečo správnejšie je zaradenie do triedy 0. Tento rozdiel je ale relatívne malý. Je to vďaka približne rovnako veľkej trénovacej množine pre obe triedy.

6.1.1.1 Graf pravdepodobného zaradenia vzorky do triedy

Lepší prehľad o tom ako pravdepodobné je zaradenie konkrétnej vzorky do jednej z množín je možné vidieť na grafe(Obr. 14) ktorý zobrazuje rovnaké údaje ako matica zmien ale navyše je v ňom aj viditeľná pravdepodobnosť zaradenia do triedy.

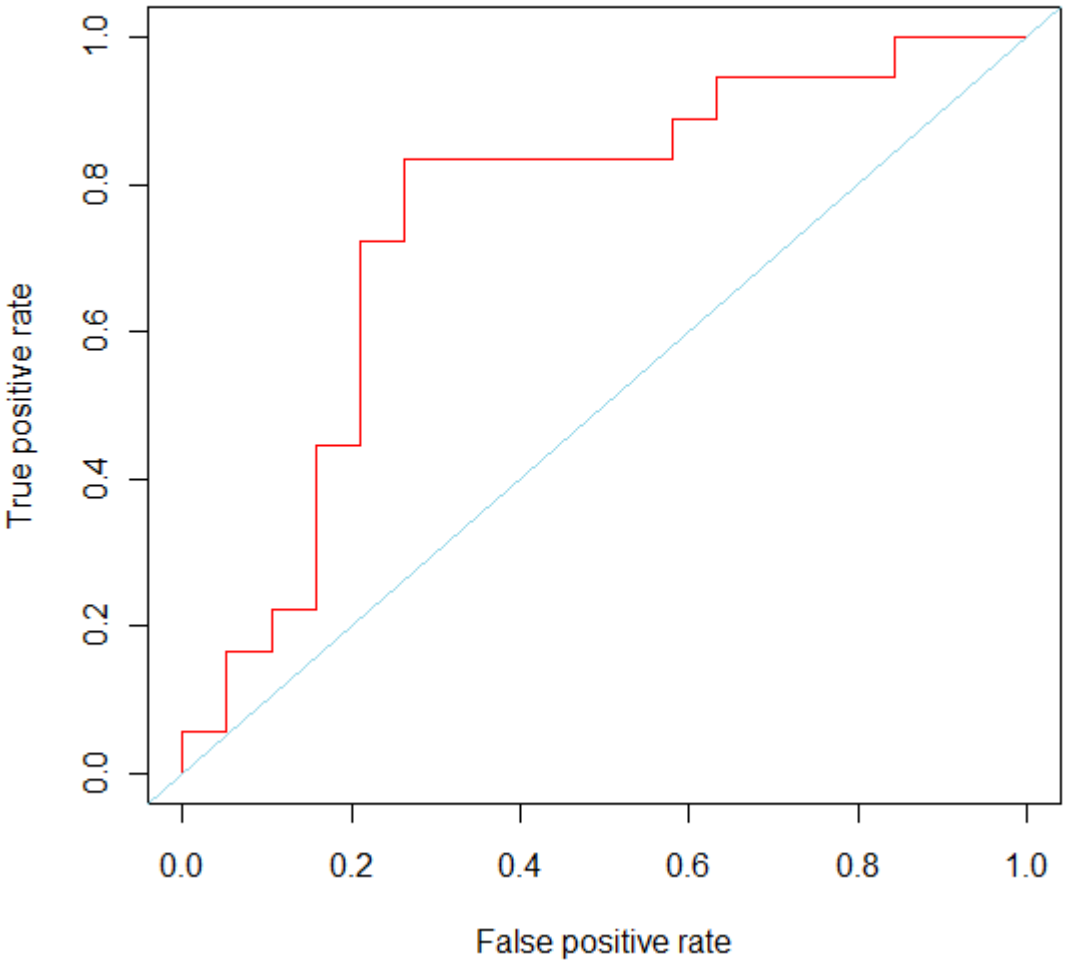


Obrázok 14

V grafe(Obr. 14) je možné vidieť v jednotlivých kvadrantoch správne a nesprávne zaradenie vzorky do triedy spolu s pravdepodobnosťou tohto zaradenia. Ľavý dolný kvadrant označuje vzorky ktoré boli správne zaradené do triedy 0. Čím bližšie je prvok k osy x, tým je väčšia pravdepodobnosť jeho zaradenia do triedy 0. Čím je prvok vzdialenejší od osy x tým je väčšia jeho pravdepodobnosť zaradenia do triedy 1. Prvky v ľavom hornom kvadrante sú označené červenou, pretože vieme, že prvých 19 vzoriek má byť správne zaradených do triedy 0, pri týchto prvkoch je väčšia pravdepodobnosť že budú zaradené do triedy 1.

6.1.1.2 ROC krivka

ROC krivka zobrazuje súvislosť medzi správne vyhodnotenými a nesprávne vyhodnotenými vzorkami. ROC krivka pre vyhodnocovaný prediktor je na obrázku(Obr. 15).



Obrázok 15

Červená čiara v grafe predstavuje priebeh ROC krivky. Diagonálna čiara v grafe predstavuje hodnoty náhodného klasifikátora. Čím viac sa ROC krivka vzďaľuje od tejto diagonálnej čiary, tým je klasifikátor kvalitnejší. Číselné vyjadrenie tejto kvality poskytuje AUC.

6.1.1.3 AUC - plocha pod ROC krivkou

AUC je skalárne vyjadrenie plochy ktorá je pod ROC krivkou. Dáva kvantitatívny pohľad na kvalitu klasifikátora. Hodnota AUC pre náš klasifikátor je AUC = 0.751462. Táto hodnota je získaná prostredníctvom balíčka ROCR. Podľa tabuľky (Tab. 2) je takýto klasifikátor hodnotený ako „dobrý“.

6.1.2 Vyhodnotenie výsledkov dosiahnutých na druhej časti datasetu

Pre vyhodnotenie výsledkov klasifikácie druhej časti datasetu som použil klasifikátor, ktorého vygenerovanie je popísané v predchádzajúcej kapitole. Je to klasifikátor, ktorý dosahoval najlepšie výsledky na tejto časti datasetu. Nájdený klasifikátor dosiahol nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **21,15%.**

- Miera správnosti pre jednotlivé triedy je Acc_1= 0,8888, Acc_0 = 0,7674
- Presnosť = 0,8888 a Úplnosť = 0, 4444
- Miera F = 0,5925
- Senzitivita = 0,4444, Špecificita = 0, 9705
- Matica zámen je v tabuľke(Tab. 9)

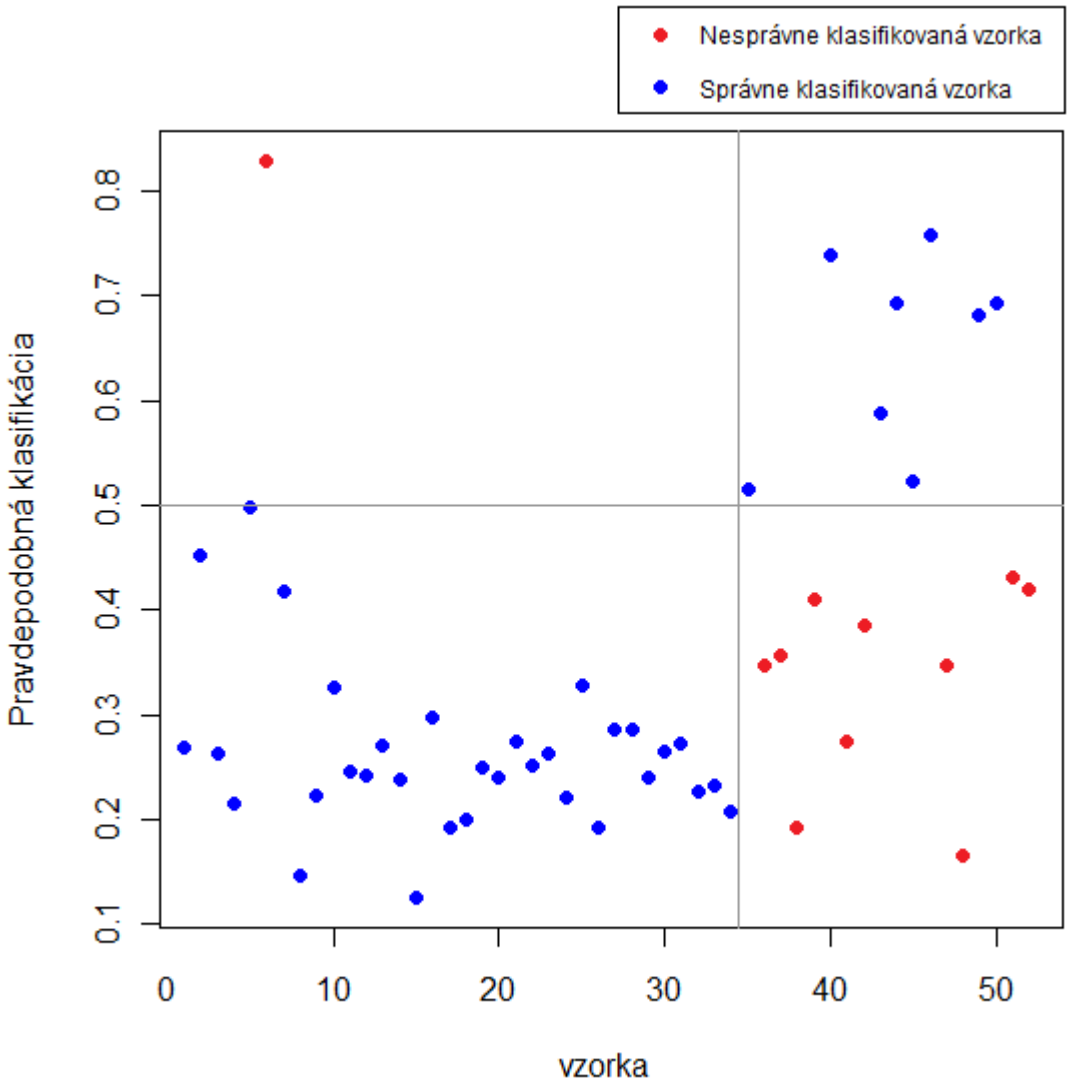
	Klasifikácia systémom	
Správne zaradenie	1	0
1	8	10
0	1	33

Tabuľka 8

Oproti klasifikátoru s prednastavenými hodnotami sa podarilo správne klasifikovať o jednu vzorku viac z každej triedy. Napriek tomu že klasifikátor zaradil do triedy 0 až 33 vzoriek správne, $Acc_0 < Acc_1$ pretože to triedy 0 zaradil až 10 vzoriek nesprávne.

6.1.2.1 Graf pravdepodobného zaradenia vzorky do triedy

Pravdepodobnosť zaradenia do triedy je v grafe (Obr. 17)

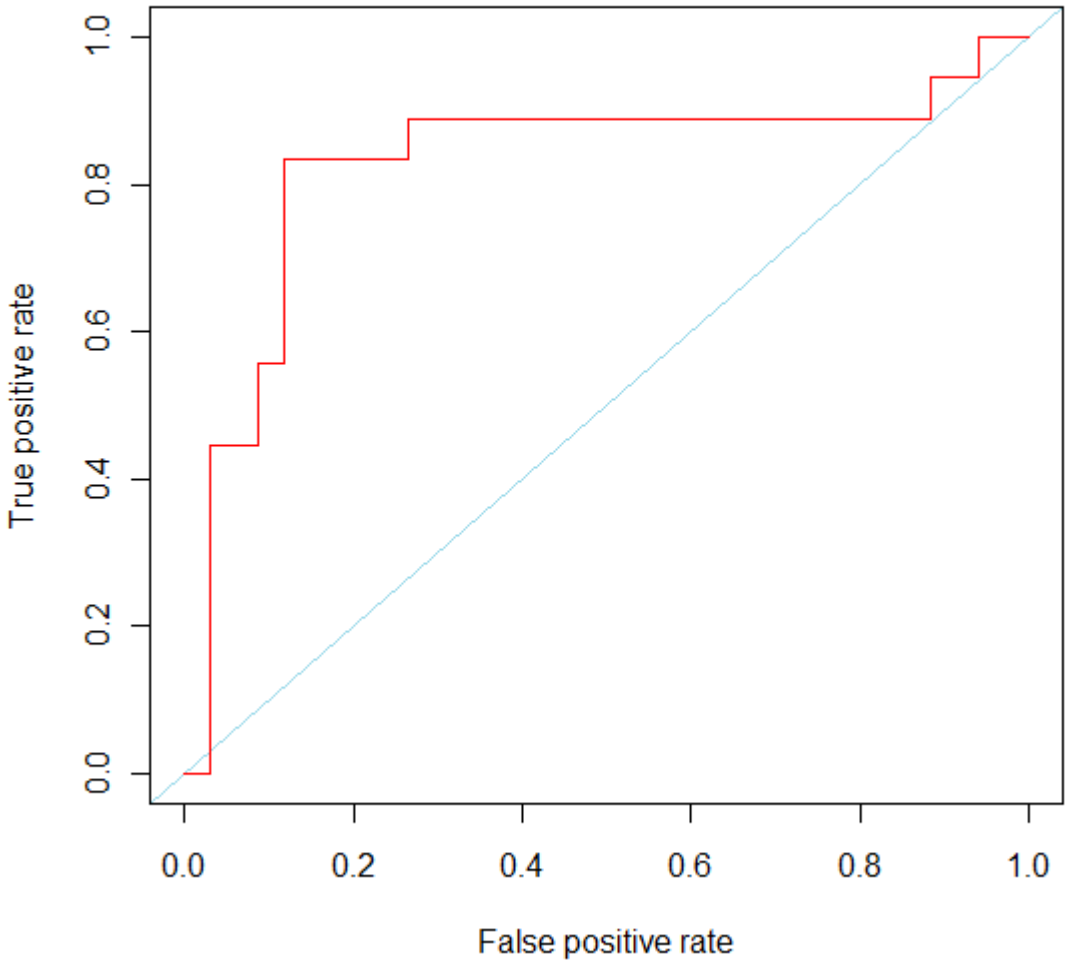


Obrázok 16

Na grafe je možné vidieť že aj keď z triedy 0 bola zaradený do nesprávnej triedy iba jedna vzorka toto zaradenie je určené s vysokou pravdepodobnosťou. Pravdepodobnostné zaradenie vzoriek z druhej časti, ktoré mali byť správne klasifikované do triedy 1 nemá takmer žiadnu vypovedajúcu hodnotu. To je spôsobené menšou trénovacou základňou triedy oproti triede 0.

6.1.2.2 ROC krivka

ROC krivka pre vyhodnocovaný prediktor je na obrázku (Obr. 18)



Obrázok 17

ROC krivka sa oproti krivke klasifikátora na prvej časti datasetu viac približuje k ideálnej hodnote v bode (0,1). Rozhodujúcou pre kvalitu klasifikácie bude plocha pod touto krivkou.

6.1.2.3 AUC - plocha pod ROC krivkou

Hodnota AUC pre testovaný klasifikátor je AUC = 0.8284314. Táto hodnota je získaná prostredníctvom balíčka ROCR. Podľa tabuľky(Tab. 2) je takýto klasifikátor hodnotený ako „veľmi dobrý“.

6.2 Vyhodnotenie metódy SVM

Keďže metóda SVM negeneruje kvalitu klasifikácie už počas trénovania ako Random Forests, pre vyhodnotenie som použil metódu leave-one-out. V 37 cykloch som teda vždy vynechal jednu vzorku a ostatné som použil pre natrénovanie klasifikátora. Následne som klasifikátorom vynechanú vzorku klasifikoval. Pre vygenerovanie pravdepodobnostného zaradenia do triedy je pre metódu SVM z balíčku e1071 nutné explicitne si takéto zaradenie vyžiadať v nastavení trénovania klasifikátora.

6.2.1 Vyhodnotenie výsledkov prvej časti datasetu

Pre vyhodnotenie výsledkov klasifikácie prvej časti datasetu som použil klasifikátory vygenerované podľa základných nastavení. Ako kernelová funkcia je použitá radiálna. Gama hodnota je nastavená na hodnotu 0,00007920792. Dosiahnuté boli nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **75,67%.**
- Miera správnosti pre jednotlivé triedy je Acc_1= 0,1111 , Acc_0 = 0,3684
- Presnosť = 0, 1111 a Úplnosť = 0, 1428
- Miera F = 0,125
- Senzitivita = 0, 1428, Špecificita = 0,3043
- Matica zámen je v tabuľke (Tab. 10)

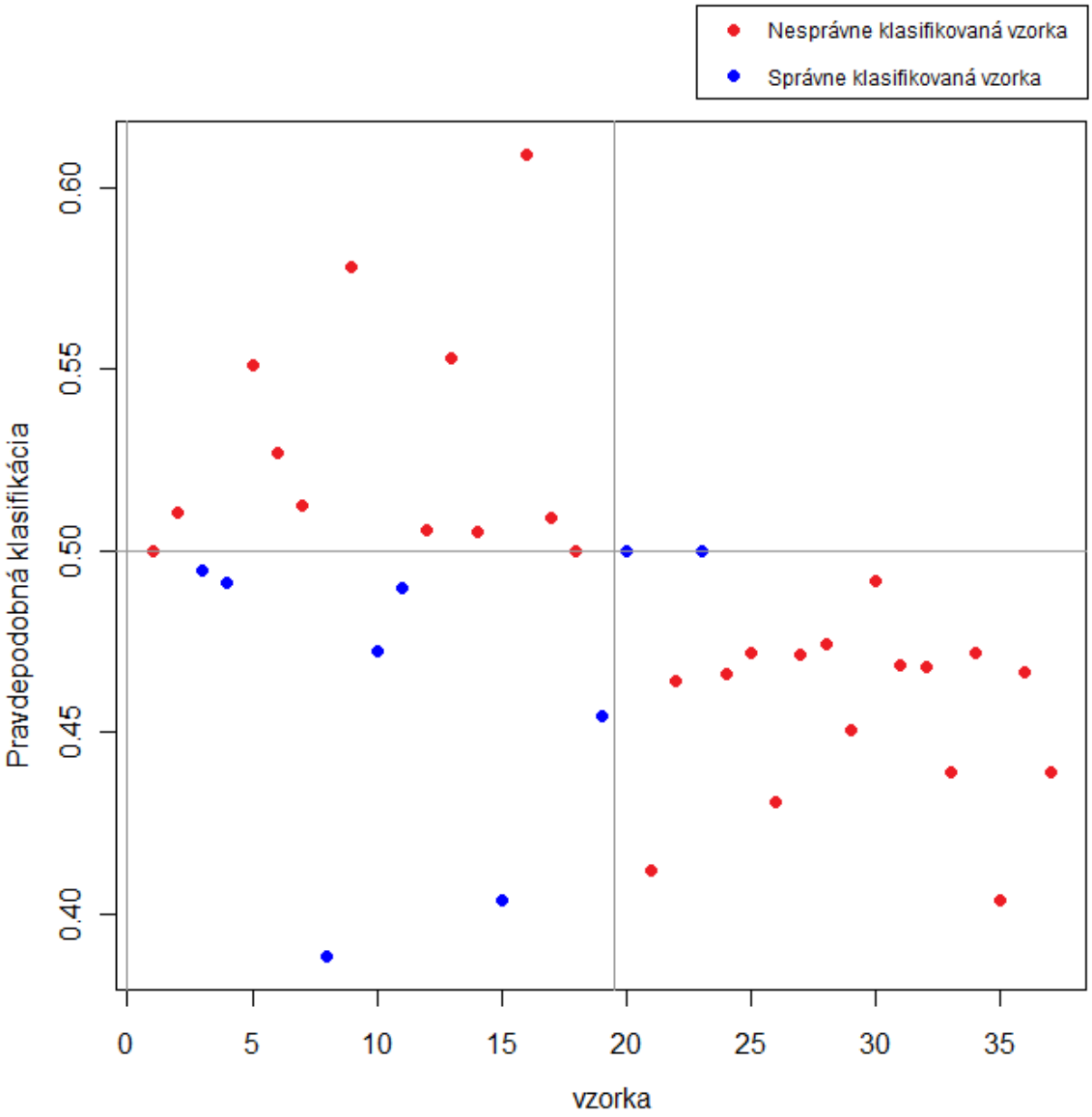
	Klasifikácia systémom	
Správne zaradenie	1	0
1	2	12
0	16	7

Tabuľka 9

Na tabuľke zámen pre jednotlivé triedy vidíme, že miera chyby klasifikátora je veľmi vysoká. Hodnota 75,67 % je dokonca vysoko nad chybovosťou náhodného klasifikátora. Najmä správnosť zaradenia do triedy 1 je až extrémne nízka. Klasifikátor takmer každú vzorku zaradil nesprávne.

6.2.1.1 Graf pravdepodobného zaradenia vzorky do triedy

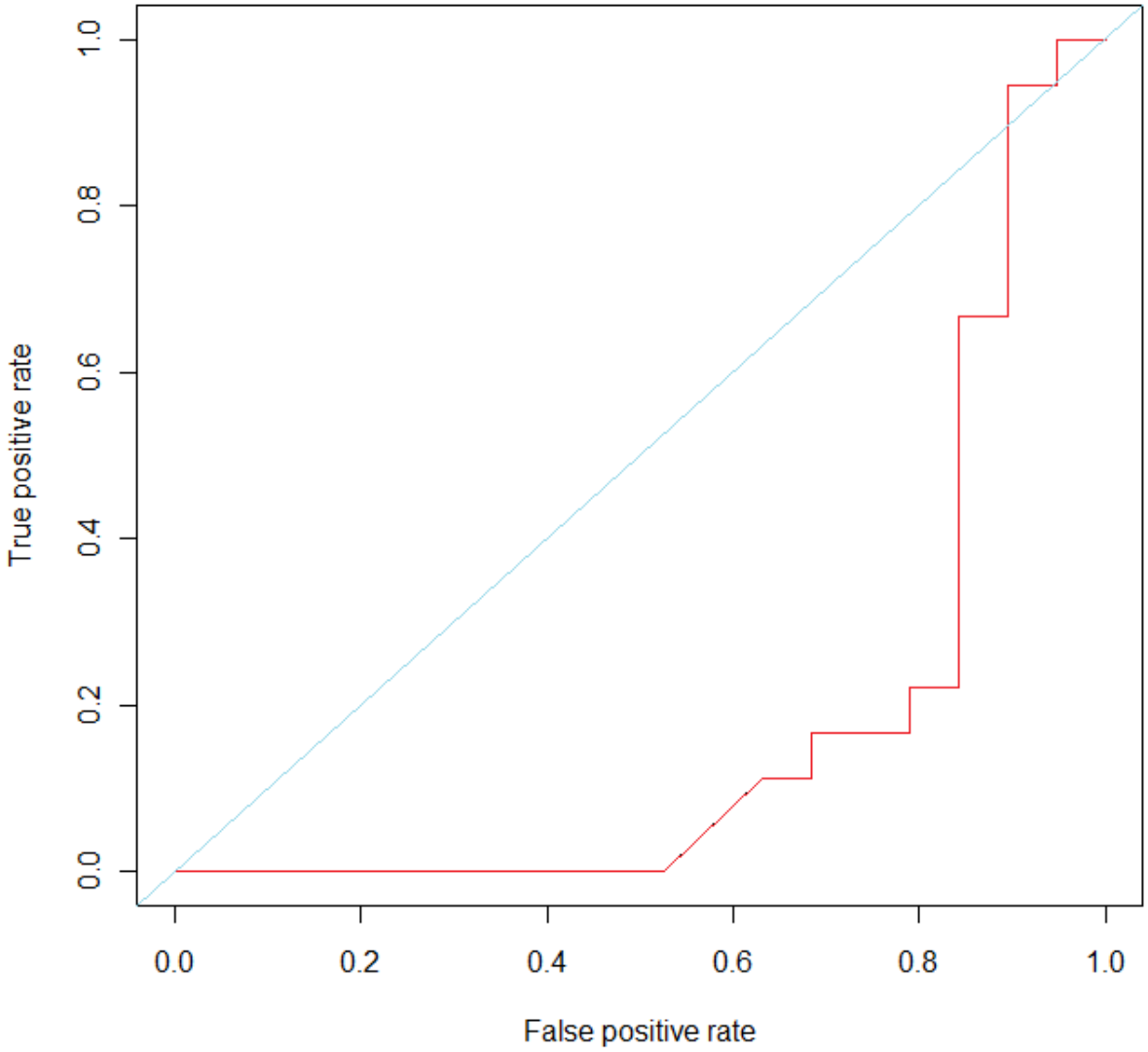
Graf pravdepodobnostného zaradenia vzoriek pre metódu SVM na prvej časti datasetu (na obr) ukazuje, ako veľmi je vygenerovaný klasifikátor nekvalitný. Dve vzorky ktoré sú jedinými vzorkami zaradenými z triedy 1 správne sú zaradené len s 50 % pravdepodobnosťou. Vzorka č.8, ktorá bola s najväčšou mierou pravdepodobnosti zaradená pri metóde Random Forests do triedy 0, dosahuje vysokú mieru aj tentokrát. Väčšina klasifikácií je však veľmi ďaleko od správneho zaradenia.



Obrázok 18

6.2.1.2 ROC krivka a AUC

ROC krivka v prípade metódy SVM vykazuje podľa očakávania hodnoty horšie ako pre náhodný klasifikátor. AUC dosahuje hodnotu 0.1783626.



Obrázok 19

6.2.2 Vyhodnotenie výsledkov na druhej časti datasetu

Pre natrénovanie klasifikátorov som použil rovnaké nastavenie ako pre prvú časť datasetu. Dosiahnuté boli nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **36,53%**.
- Miera správnosti pre jednotlivé triedy je Acc_1= 0,4736, Acc_0 = 0,7272
- Presnosť = 0,4736 a Úplnosť = 0,5
- Miera F = 0,3529
- Senzitivita = 0,5, Špecificita = 0,7058
- Matica zámen je v tabuľke (Tab. 11)

	Klasifikácia systémom	
Správne zaradenie	1	0

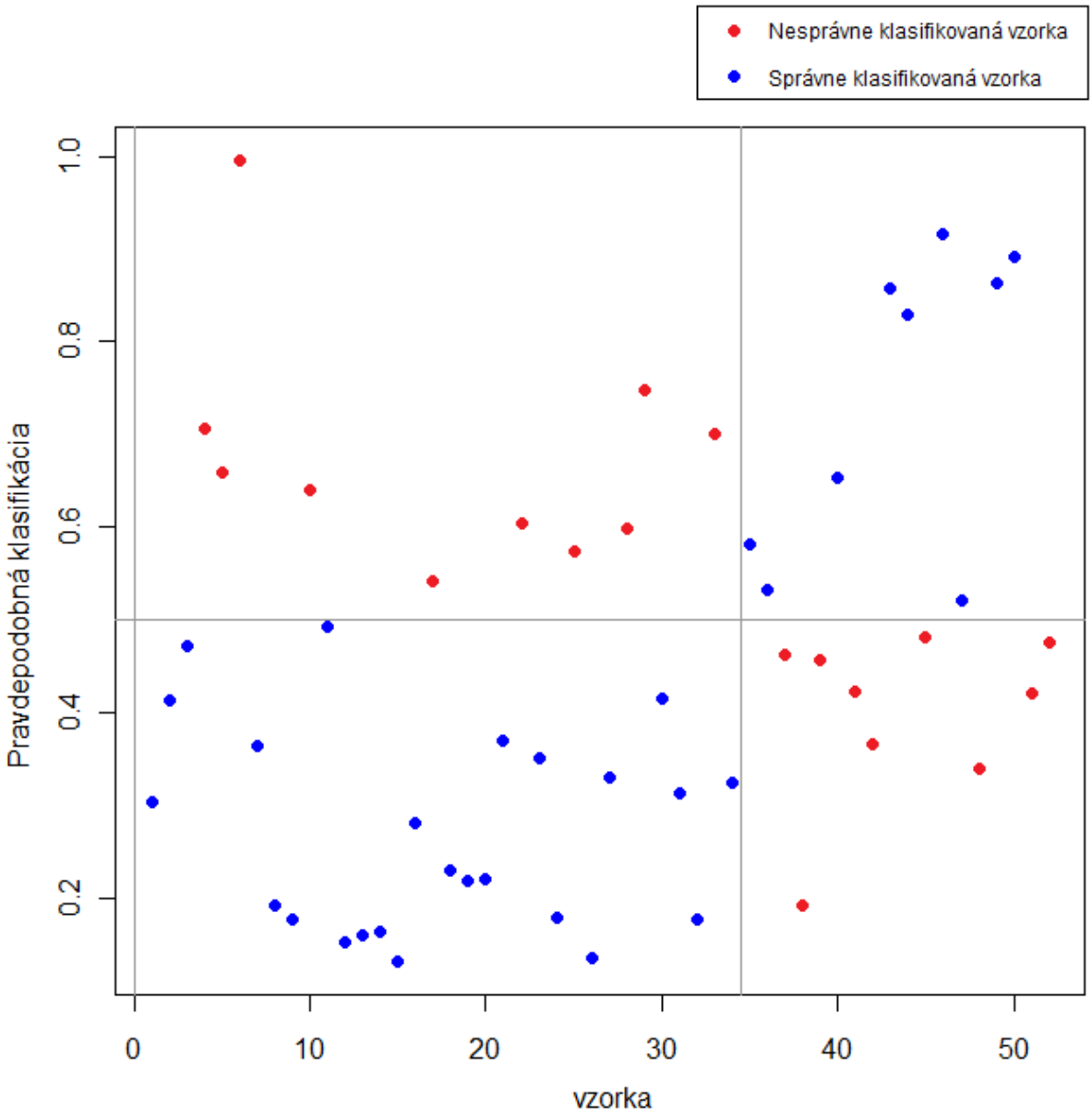
1	9	9
0	10	24

Tabuľka 10

Z tabuľky zámen pre jednotlivé triedy je vidieť, že miera chyby klasifikátora je výrazne nižšia ako v prípade metódy SVM použitej na prvej časti datasetu. Správne je zaradenie najme do triedy 0, pre ktorú je k dispozícií väčší počet vzoriek.

6.2.2.1 Graf pravdepodobného zaradenia vzorky do triedy

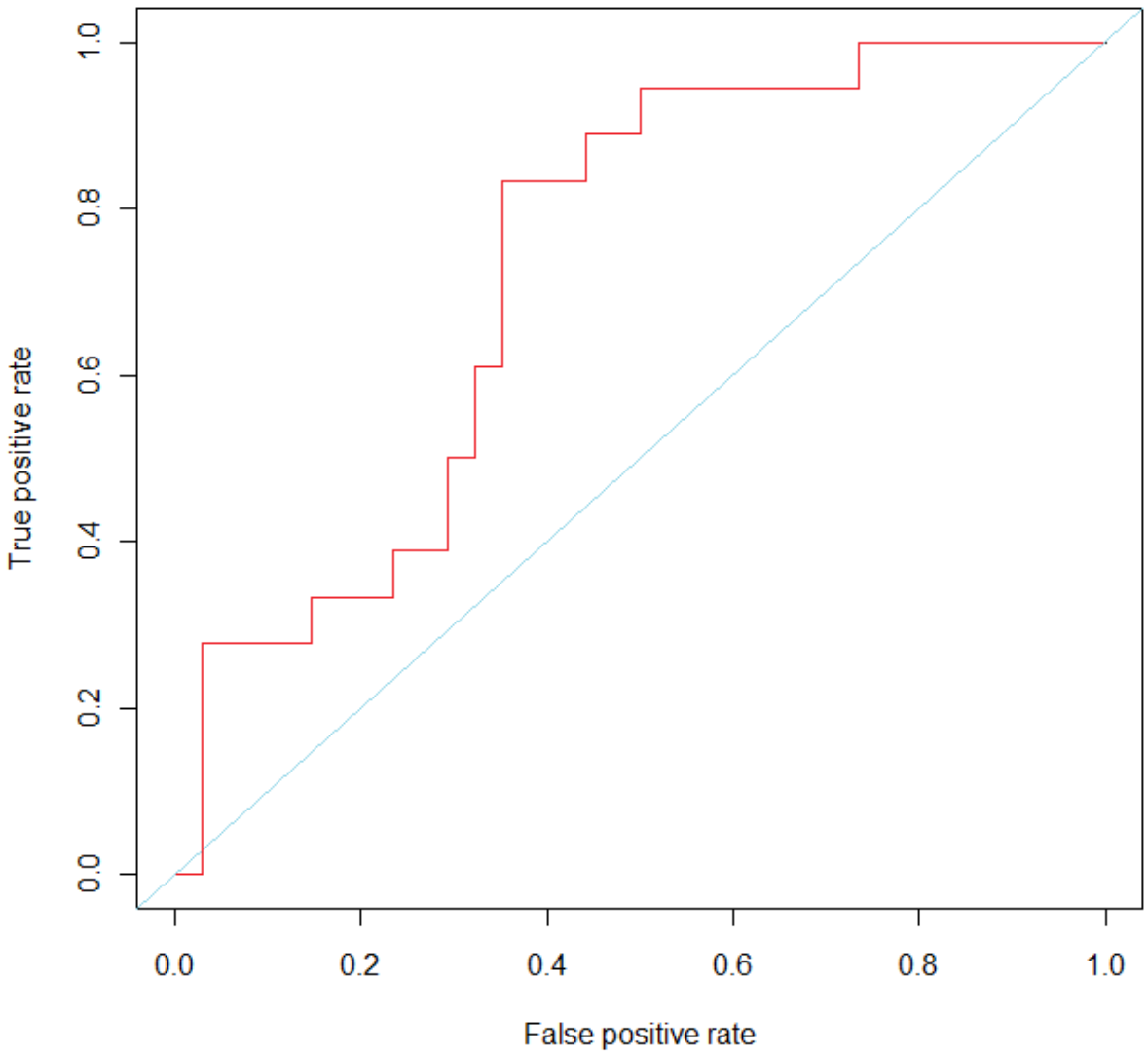
Obor hodnôt grafu pre mieru pravdepodobného zaradenia je v prípade druhej časti datasetu omnoho širší ako v prípade prvej časti. Vzorka číslo 6 je podobne ako aj v prípade metódy Random Forests klasifikovaná správne s takmer 100% pravdepodobnosťou. V prípadoch zaradenia prvkov z triedy 1 vyzerá graf takmer náhodne. Nesprávne zaradené vzorky dosahujú vyššiu mieru pravdepodobnosti zaradenia ako tie správne zaradené.



Obrázok 20

6.2.2.2 ROC krivka a AUC

ROC krivka vyzerá oproti prvej časti datasetu omnoho optimistickejšie. Je to najmä vďaka správnemu zaradeniu vzoriek z triedy 0. Aj tak ale nedosahuje výsledky ako v prípade metódy Random Forests. Plocha pod krivkou je AUC = 0.7303922. Podľa tabuľky (Tab. 2) je takýto klasifikátor hodnotený ako „dobrý“.



Obrázok 21

6.3 Vyhodnotenie metódy Rozhodovací strom

Pre vyhodnotenie klasifikácií som použil takisto ako v prípade SVM testovanie metódou leave-one-out. V nastavení klasifikácie som implicitne nastavil vytváranie klasifikačných stromov. Stromy sa neorezávajú. Pre správne vyhodnotenie klasifikácie je potrebné poznať pravdepodobnostné miery klasifikácie. Zvolená metóda ale na datasete skúmanom v tejto práci generuje pomerne malé stromy, ktoré majú v priemere 5 uzlov. Klasifikácia do triedy je teda často absolútna 1, alebo 0.

6.3.1 Vyhodnotenie výsledkov prvej časti datasetu

Na prvej časti datasetu boli dosiahnuté nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **27,02%**.
- Miera správnosti pre jednotlivé triedy je Acc_1= 0,7222 , Acc_0 = 0,7368

- Presnosť = 0,7222 a Úplnosť = 0,7222
- Miera F = 0,72222
- Senzitivita = 0,7222, Špecificita = 0,7368
- Matica zámen je v tabuľke (Tab. 12)

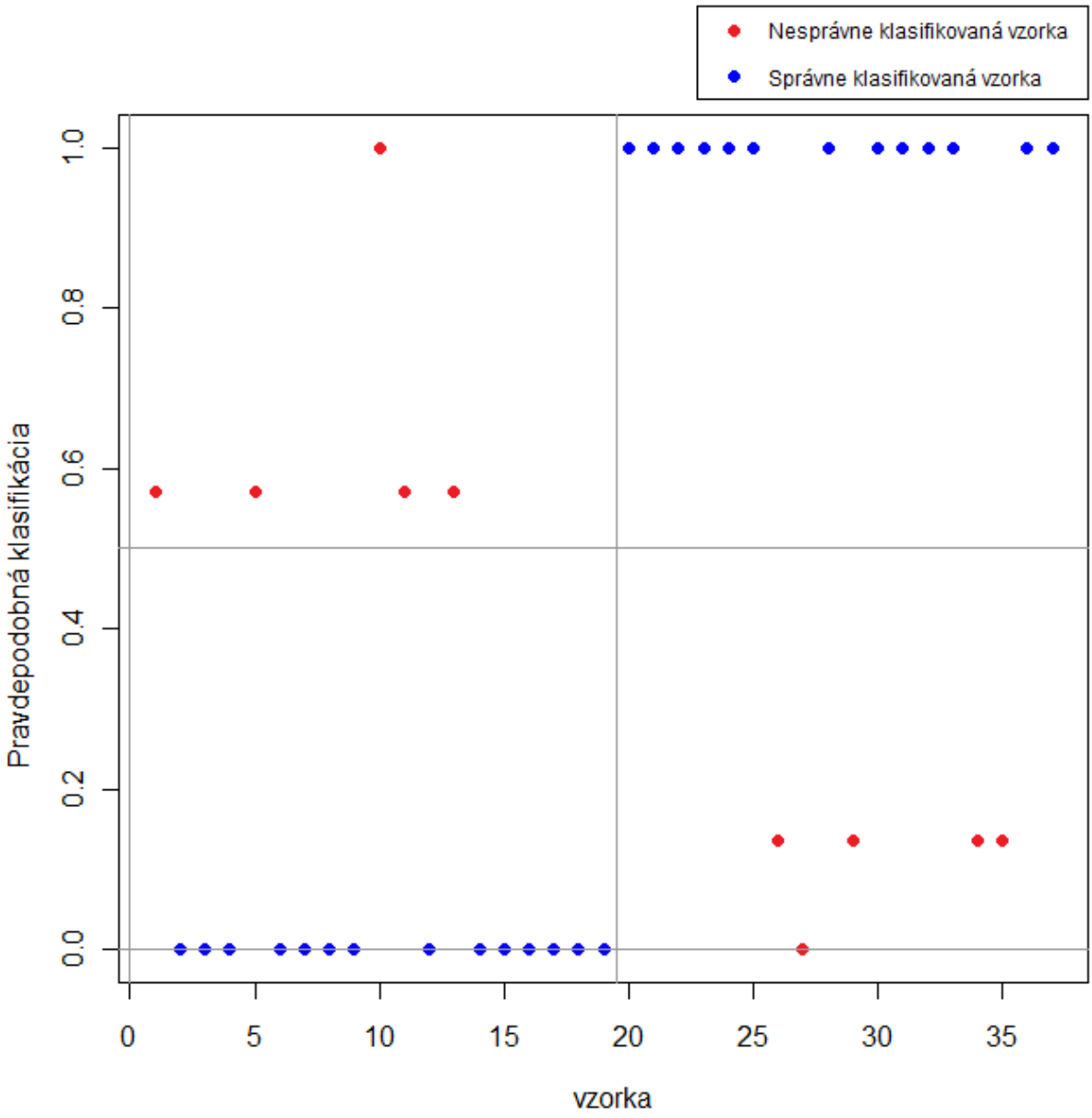
	Klasifikácia systémom	
Správne zaradenie	1	0
1	13	5
0	5	14

Tabuľka 11

Napriek tomu že metóda vytvára jednoduché stromy, dosahuje na prvej časti datasetu veľmi dobré výsledky. Nedosahuje úroveň akú dosiahla Random Forests, ale pre obe triedy má vyvážené výsledky.

6.3.1.1 Graf pravdepodobného zaradenia vzorky do triedy

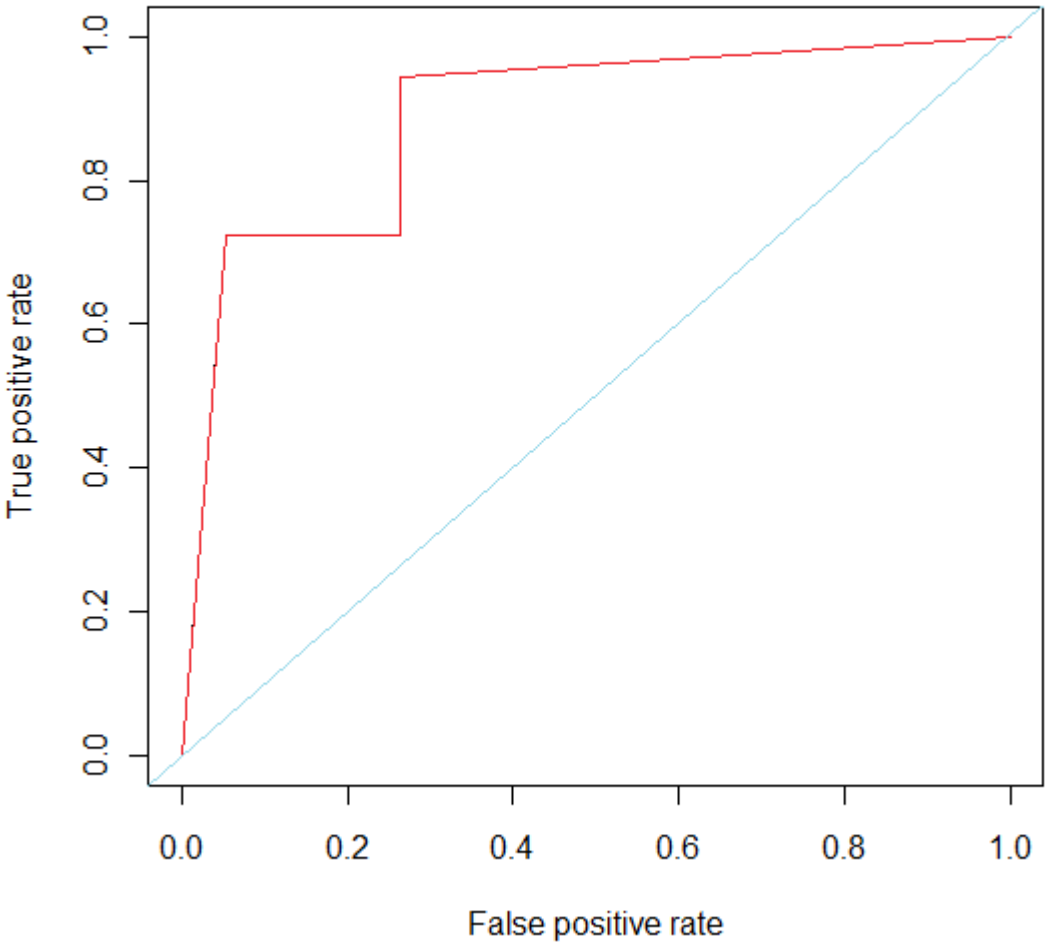
Na grafe je vidieť ako sú vzorky sú klasifikované iba do štyroch úrovní. Úplne 1, úplne 0, približne 1, približne 0. Správne zaradené vzorky sú spravidla zaradené so 100 % pravdepodobnosťou. V dvoch prípadoch je ale aj 100 % pravdepodobnosť nesprávneho zaradenia vzorky.



Obrázok 22

6.3.1.2 ROC krivka a AUC

Ako aj graf pravdepodobnostného rozloženia, aj ROC krivka má veľmi strohý charakter. Je ale evidentne vysoko nad úrovňou náhodného klasifikátora. Plocha pod krivkou dosahuje hodnotu AUC = 0.8874269, čo je jednoznačne najvyššia dosiahnutá hodnota z nami skúmaných klasifikátorov.



Obrázok 23

6.3.2 Vyhodnotenie výsledkov na druhej časti datasetu

Pre druhú časť datasetu boli nastavené rovnaké parametre ako pre prvú časť. Boli dosiahnuté nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **50%.**
- Miera správnosti pre jednotlivé triedy je Acc_1= 0,3 , Acc_0 = 0,625
- Presnosť = 0,3 a Úplnosť = 0,3333
- Miera F = 0,3157
- Senzitivita = 0,3333, Špecificita = 0,5882
- Matica zámen je v tabuľke (Tab. 13)

	Klasifikácia systémom	
Správne zaradenie	1	0
1	6	12

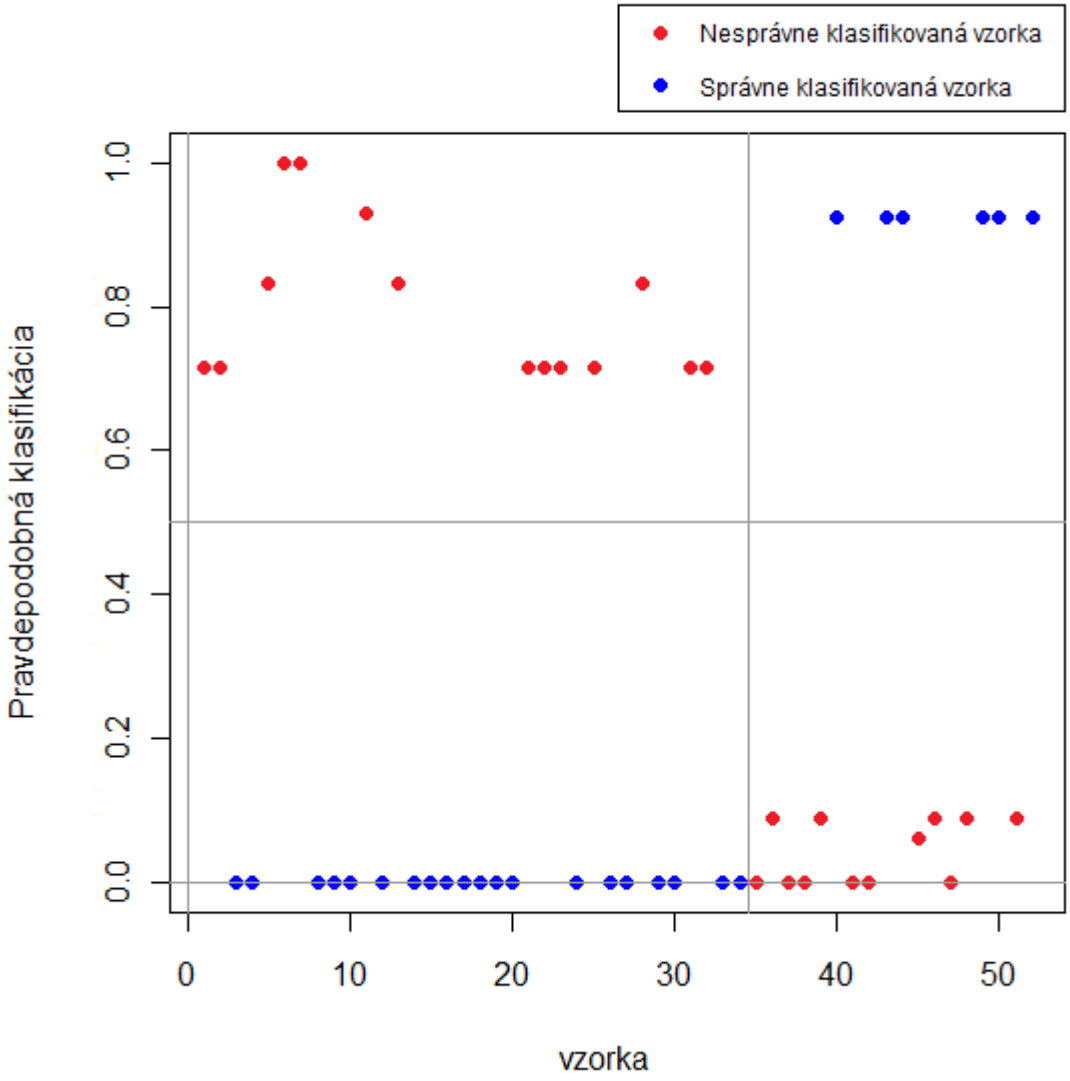
0	14	20

Tabuľka 12

Na druhej časti datasetu už metóda nedosahuje až také dobré výsledky. Správne je zaradená iba polovica vzoriek, čo zaraduje metódu na úroveň náhodného klasifikátora. Miera správnosti je nízka najmä pre zaradenie do triedy 1.

6.3.2.1 Graf pravdepodobného zaradenia vzorky do triedy

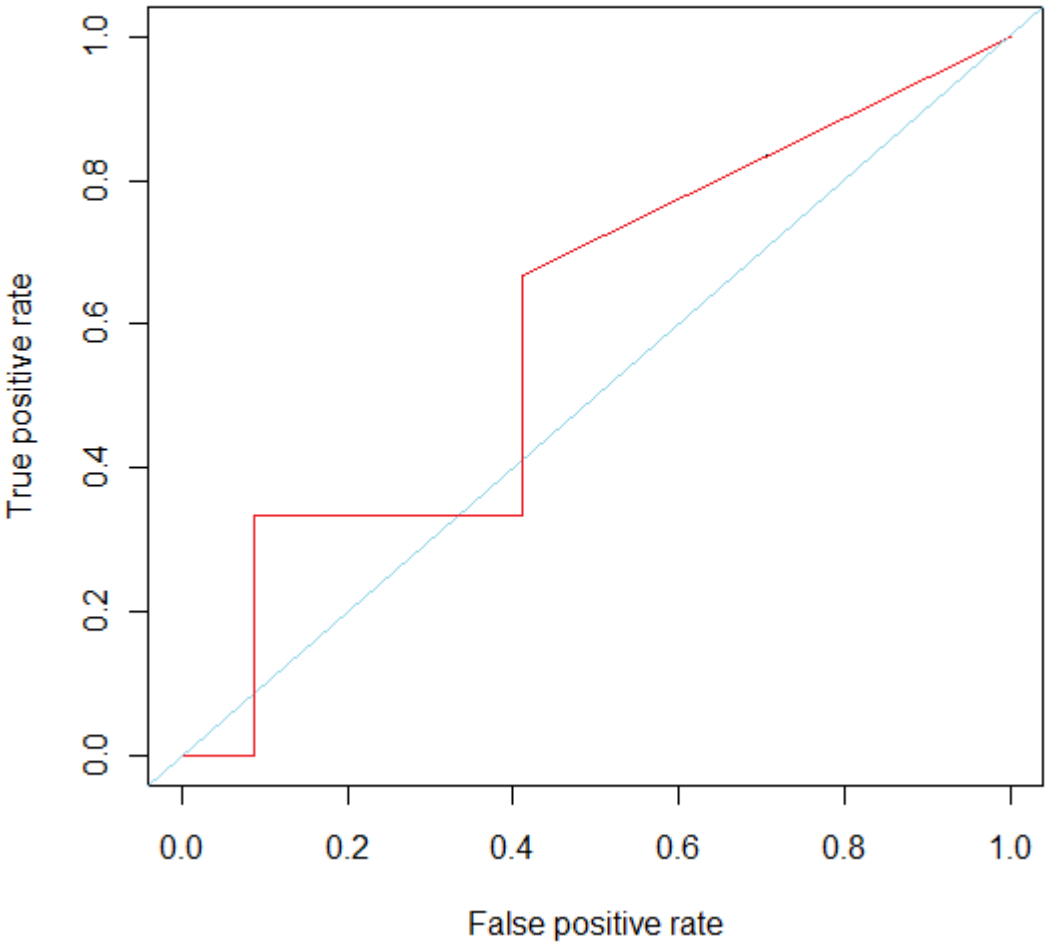
Na grafe vidíme, že ako aj pri klasifikácií na prvej časti datasetu aj v tomto prípade každé správne zaradenie je s vyskokou mierou pravdepodobnosti. Pri klasifikácií do triedy 0 je až 20 vzoriek klasifikovaných správne so 100 % pravdepodobnosťou. Rozloženie už nie je až tak strohé, vďaka väčšiemu počtu trénovacích vzoriek.



Obrázok 24

6.3.2.2 ROC krivka a AUC

ROC krivka má podobný priebeh ako na prvej časti datasetu, ale na rozdiel od nej sa viac približuje k osi pred predstavujúcej náhodný klasifikátor. Tomu zodpovedá aj hodnota AUC = 0.5980392



Obrázok 25

6.4 Vyhodnotenie metódy Neurónová sieť

Metódu Neurónové siete som vyhodnocoval iba na prvej časti datasetu. Pre trénovanie bola použitá neurónová sieť s jednou skrytou vrstvou v ktorej boli tri neuróny. Pretože do neurónov v skrytej vrstve vedie váhované spojenie z každého vstupu, je potrebné implicitne nastaviť maximálny počet váh na 3*12625, pretože 12625 prvkov má každá vzorka v datasete. Pre vyhodnotenie bola opäť použitá metóda leave-one-out.

6.4.1 Vyhodnotenie výsledkov prvej časti datasetu

Na prvej časti datasetu boli dosiahnuté nasledovné výsledky:

- Dosiahnutý odhad miery chyby je na úrovni **51,35%.**
- Miera správnosti pre jednotlivé triedy je Acc_1= 0,4666 , Acc_0 = 0,5
- Presnosť = 0,4666 a Úplnosť = 0,3888
- Miera F = 0,4242
- Senzitivita = 0,3888, Špecificita = 0,5789

- Matica zámen je v tabuľke (Tab. 14)

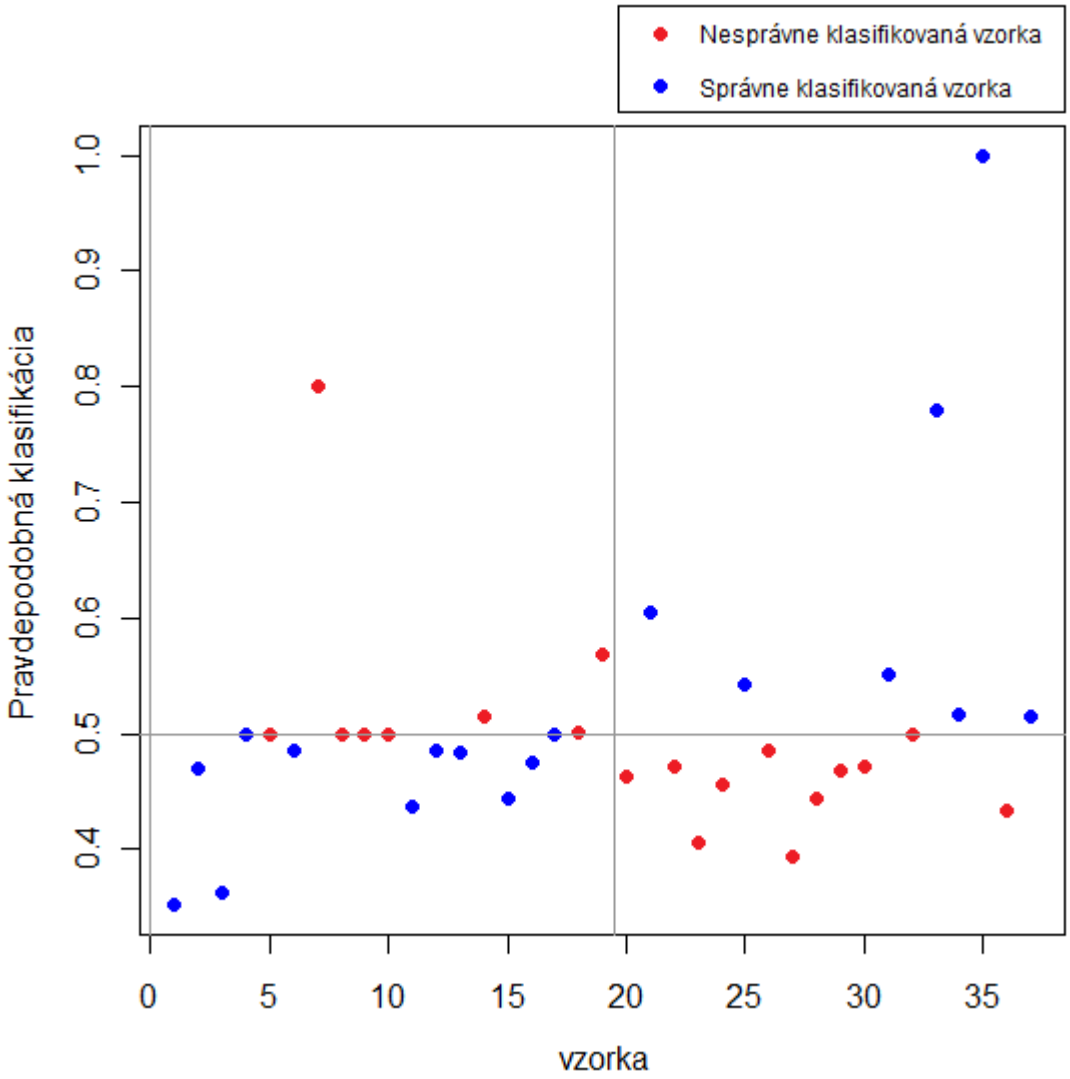
	Klasifikácia systémom	
Správne zaradenie	1	0
1	7	11
0	8	11

Tabuľka 13

Výsledná dosiahnutá miera chyby je až vyše 50%. Táto chybovosť približne platí pre zaradenie do oboch tried.

6.4.1.1 Graf pravdepodobného zaradenia vzorky do triedy

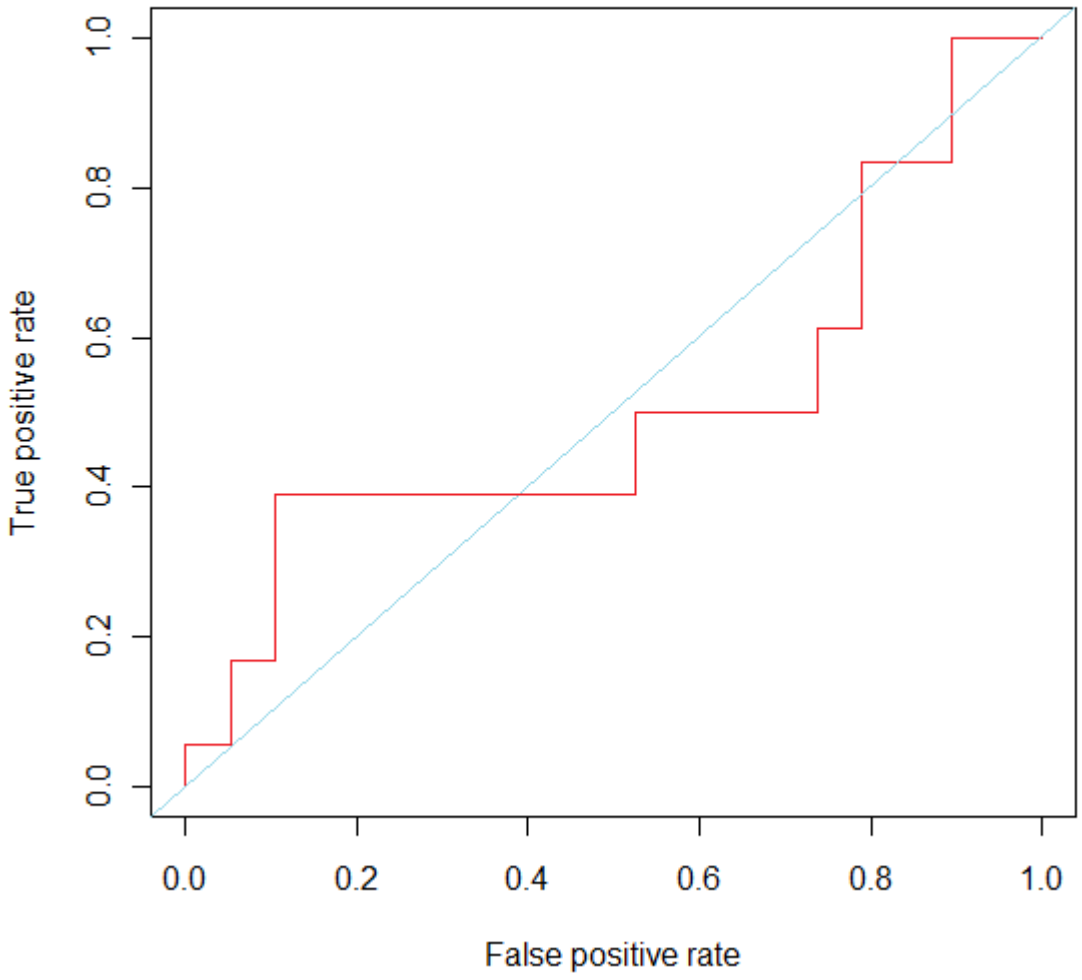
Pravdepodonostná miera klasifikácie ukazuje ako sa klasifikácie veľmi blížia k 50% miere zaradenia. Z toho vyplýva aj miera chyby s takouto hodnotou. Takéto rozloženie naznačuje pretrénovanie klasifikátorov.



Obrázok 26

6.4.1.2 ROC krivka a AUC

Pretrénovanie sa odzrkadľuje aj na tvare ROC krivky, ktorá sa len minimálne vzďaľuje od diagonálnej osi. Hodnota AUC = 0.505848.



Obrázok 27

7 Zhodnotenie a porovnanie výsledkov

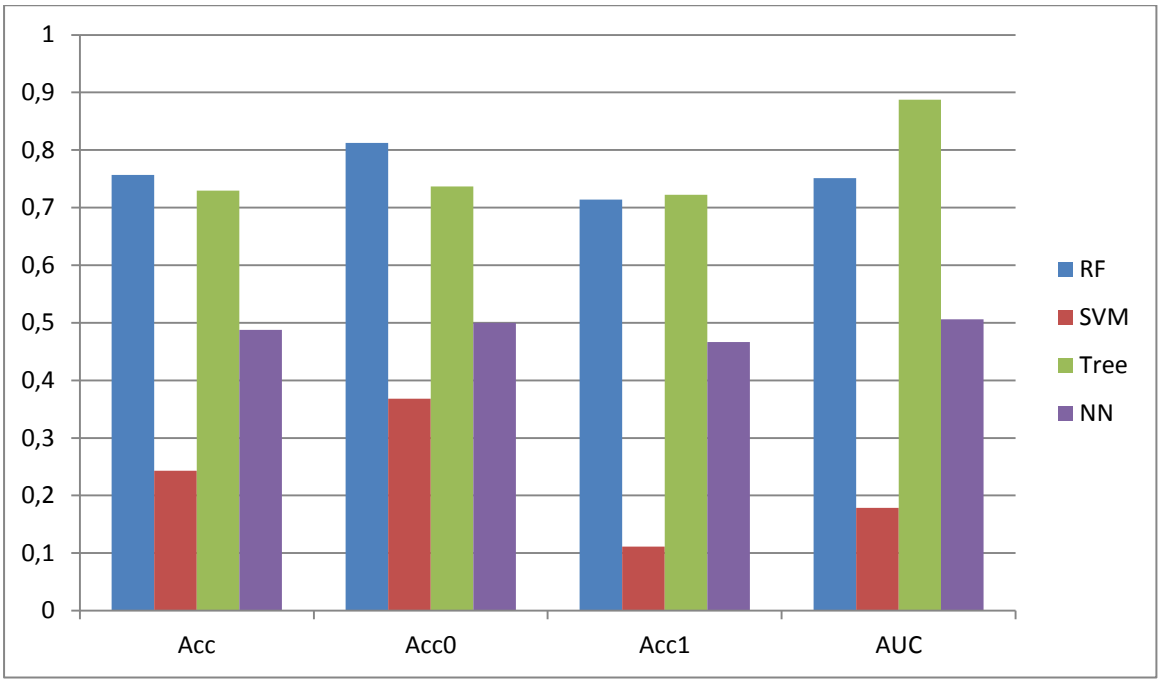
7.1 Zhodnotenie výsledkov podľa metód

Pre zhodnotenie metód som použil 4 hodnoty ktoré určujú kvalitu klasifikácie:

- Celková správnosť klasifikácie Acc
- Správnosť klasifikácie pre triedu 0, Acc_0
- Správnosť klasifikácie pre triedu 1, Acc_0
- Plocha pod ROC krivkou, AUC

7.1.1 Zhodnotenie výsledkov na prvej časti datasetu

Všetky výsledné hodnoty sú zobrazené na grafe.

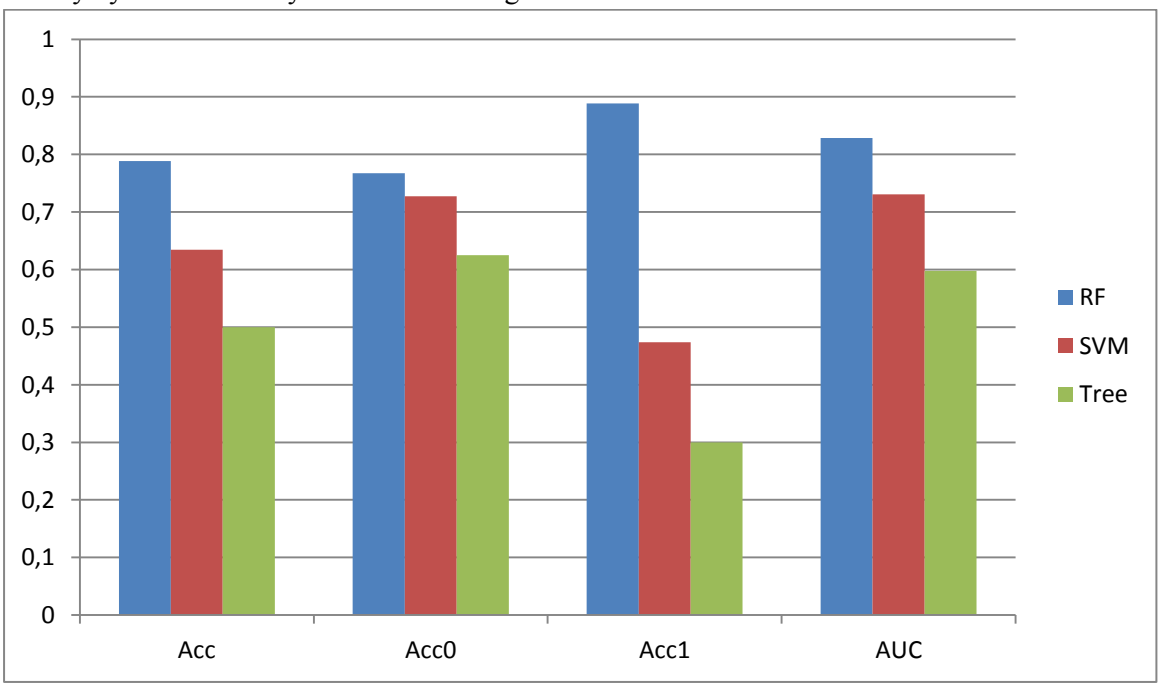


Obrázok 28

Najlepšie v porovnaní vychádzajú metódy Random Forests a Klasifikačný strom. Najmä hodnota AUC takmer 0,9 je veľmi dobrý výsledok. Neurónové siete sa vo všetkých ukazovateľoch pohybujú v blízkosti hodnoty 0,5. Napriek najvyšším pamäťovým a výpočtovým nárokom neponúkajú generované klasifikátory takmer žiadnu výpovednú hodnotu. Rovnako zle dopadla metóda SVM ktorá výraznú väčšinu vzoriek klasifikovala nesprávne.

7.1.2 Zhodnotenie výsledkov na druhej časti datasetu

Všetky výsledné hodnoty sú zobrazené na grafe.



Obrázok 29

Ako aj na prvej časti datasetu aj teraz dosiahla metóda Random Forest kvalitné výsledky, Najvyššie vo všetkých štyroch meraniach. Na druhom mieste sa nachádza vo všetkých meraniach metóda SVM. Najhoršie dopadla metóda Klasifikačný strom. Priemerne najkvalitnejšie výsledky sú dosiahnuté pre správnosť zaradenia do triedy 0, pre ktorú bolo v datasete viacero vzoriek ako pre triedu 1.

7.2 Zhodnotenie výsledkov metódy Random Forests

Výsledky som zhodnotil podľa hodnôt dosiahnutých najlepšími klasifikátormi vygenerovanými na prvej a druhej časti datasetu.

Celkovo môžeme dosiahnuté výsledky považovať za priemerné. V porovnaní síce dopadli horšie, ale správnosť klasifikácie možno hodnotiť ako uspokojivú.

7.2.1 Zhodnotenie výsledkov dosiahnutých na prvej časti datasetu

Výsledky dosiahnuté na prvej časti datasetu sa dajú považovať za kvalitné na úrovni približne 75%. (miera chyby 25%).

Klasifikácia do oboch tried vykazuje porovnateľnú chybovosť. Pravdepodobnostné zaradenie vzoriek do tried však ukazuje, že aj správna klasifikácia nie je veľmi vzdialená od nesprávnej a vyplýva z toho riziko, že môže byť zaradená do nesprávnej triedy. Stavový priestor nieje dostatočne rozdelený na to, aby mohli byť výsledky klasifikácie iných ako trénovacích dát považované za relevantné.

7.2.2 Zhodnotenie výsledkov dosiahnutých na druhej časti datasetu

Výsledky dosiahnuté na druhej časti datasetu sa dajú považovať podľa miery chybovosti za kvalitné na približne 80%.

Ak sa však pozeráme na klasifikáciu podľa skutočných tried, zistíme, že pre triedu 1, čiže prípady kedy sa rakovina opäť prejavila nie je klasifikátor takmer vôbec schopný predikovať. Výsledky na tejto časti sú približne rovnaké ako s náhodným klasifikátorom. Naproti tomu vzorky zo skutočnej triedy 0 boli klasifikované s vysokou mierou úspešnosti. Na obrázku pravdepodobnej klasifikácie je možné vidieť že stavový priestor v tejto oblasti je velmi dobre rozdelený. To je aj dôvod, prečo je hodnota AUC v tomto prípade vyššia.

7.3 Porovnanie výsledkov s dostupnými výsledkami

Pre porovnanie dosiahnutých výsledkom som použil prácu [3]. Autori v nej popisujú svoje výsledky klasifikácie nad tým istým datasetom ktorý som použil ja. Ako klasifikátor použili stromový model ktorý nijak bližšie nešpecifikovali. Dáta predspracovávajú pomocou zhlukovej analýzy, na základe ktorej z trénovacích dát odstránia gény ktoré vykazovali nízku variáciu. Pomocou tejto metódy dosiahli až 90% úspešnosť predikcie.

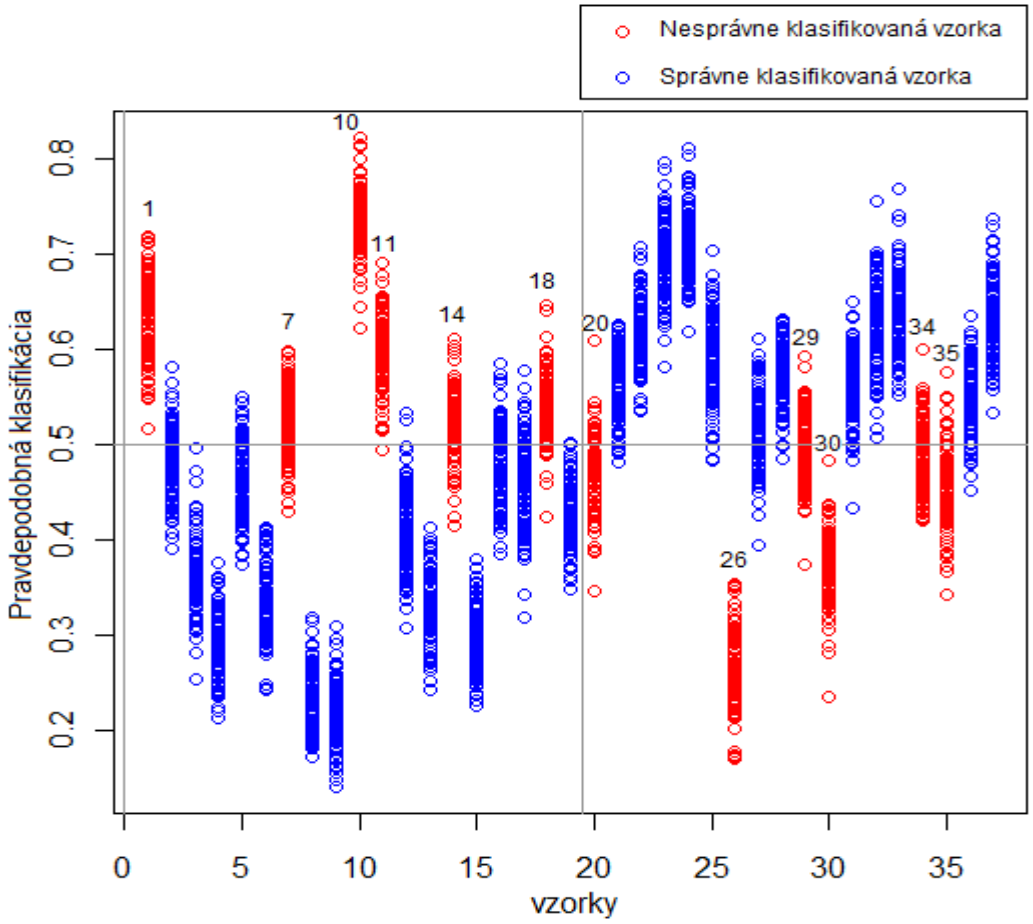
7.3.1 Porovnanie výsledkov dosiahnutých na prvej časti datasetu

Pre porovnanie výsledkov som vytvoril graf (Obr. 15), ktorý má podobný charakter ako graf publikovaný v porovnávanej práci. Tento graf som vytvoril na základe pravdepodobných klasifikácií generovaných klasifikátormi s parametrom mtry v rozsahu 100 až 10 000. V grafe je zobrazený celý rozsah pravdepodobnej klasifikácie pre všetky generované klasifikátory. Pre lepšiu prehľadnosť sú vzorky, ktoré majú väčšinu časť pravdepodobnej klasifikácie v nesprávnej triede označené červenou farbou a očíslované.

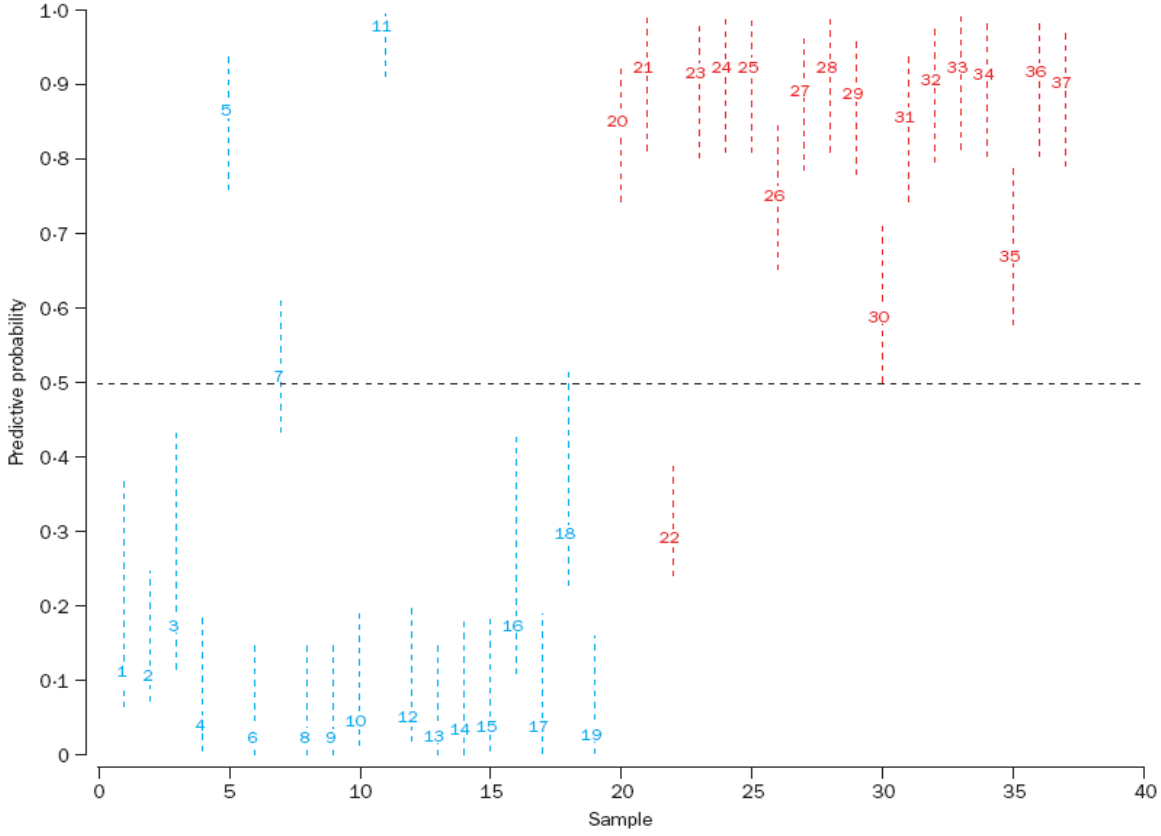
Graf(Obr. 16) z porovnávanej publikácie takisto zobrazuje pravdepodobné zaradenie do tried. Testovanie v tomto prípade vykonali pomocou krížovej validácie. Modrou farbou sú v grafe označené vzorky, ktoré majú byť správne zaradené do triedy 0 a červenou farbou vzorky ktoré majú byť zaradené do triedy 1.

V porovnaní výsledkov vychádza klasifikátor porovnávanej práce jednoznačne lepšie. Nielen že správne klasifikoval väčší počet vzoriek, ale zároveň je pravdepodobnosť zaradenia vzorky do správnej triedy vo väčšine prípadov veľmi vysoká.

Napriek tomu však je možné nájsť vzorky (č.5 a č.22), ktoré náš klasifikátor, na rozdiel od porovnávaného, zaradil do správnej triedy. Aj tak je však z celkového pohľadu porovnávaný klasifikátor lepší.



Obrázok 30



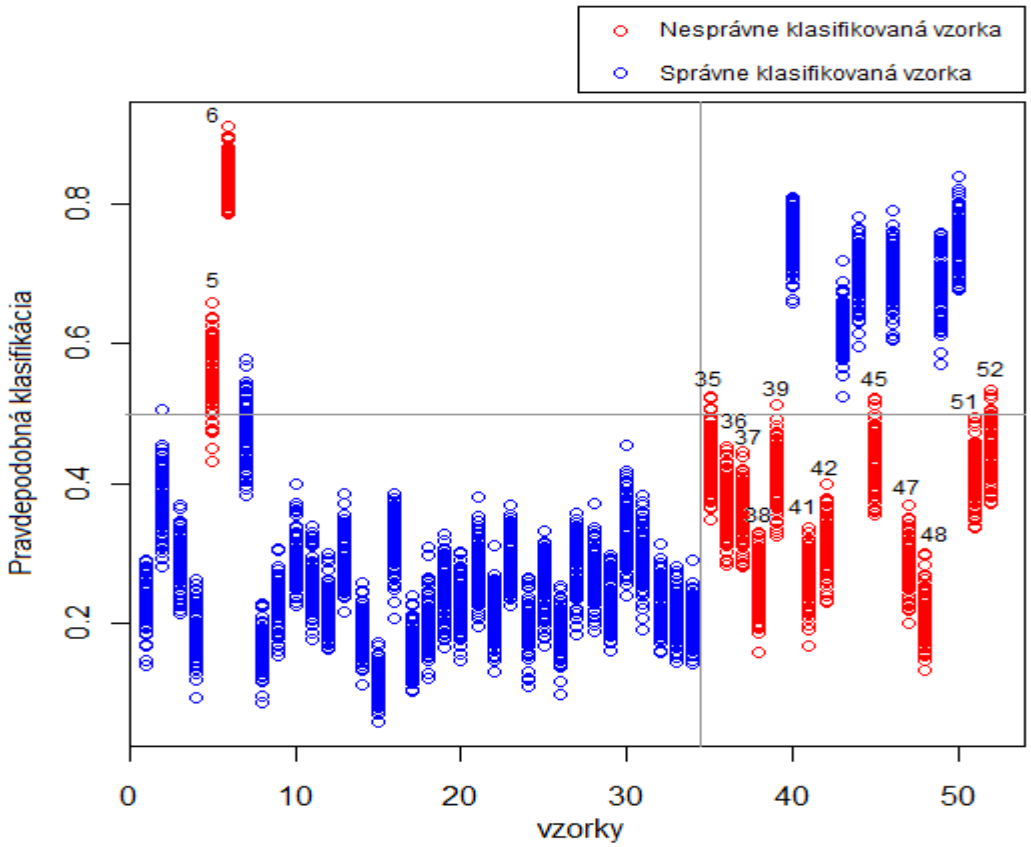
Obrázok 31

7.3.2 Porovnanie výsledkov dosiahnutých na druhej časti datasetu

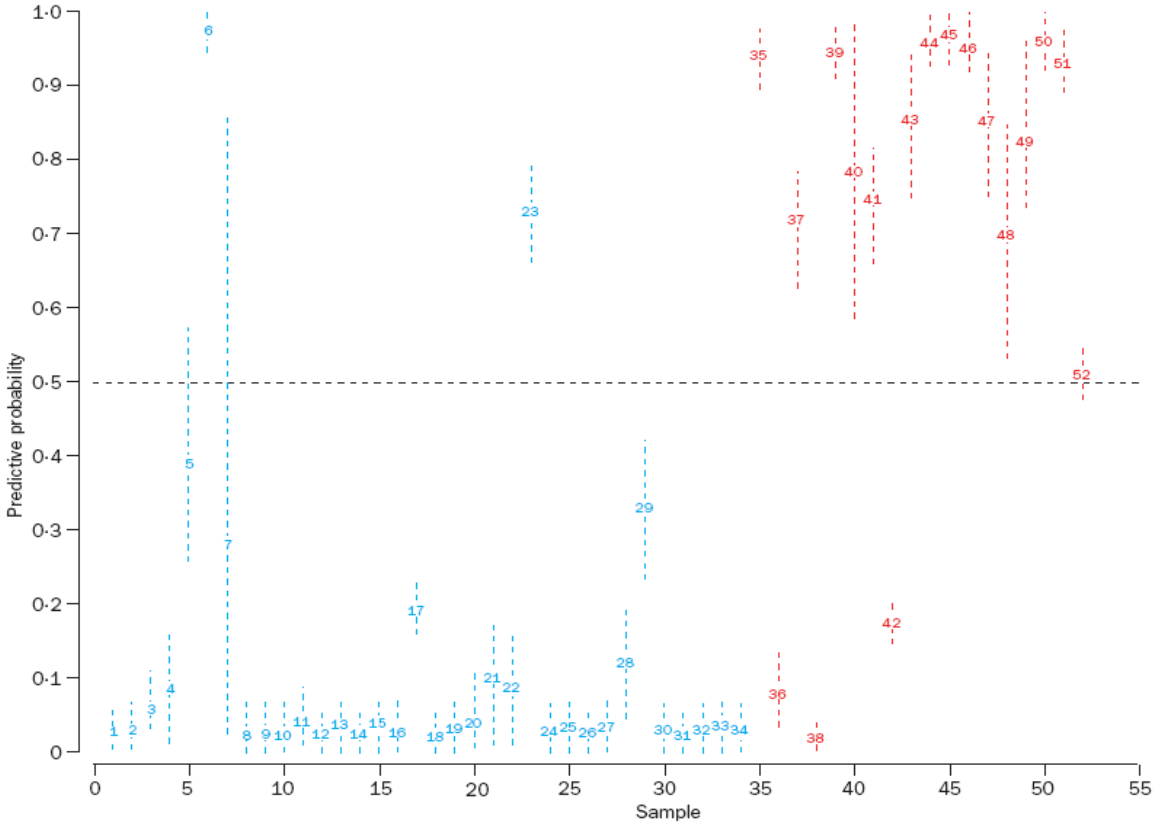
Porovnávané grafy(Obr. 17,Obr. 18) sú obdobé ako pre prvú časť datasetu.

Rozdiely v grafoch je možné pozorovať najmä v zaradení objektov triedy 1. Kým v našom prípade sú tieto objekty rozdelené na úrovni náhodného klasifikátora, v prípade porovnávaného grafu sú zaradené do správnej triedy vo veľkej väčšine prípadov. Metóda ktorú použili sa s týmto nebezpečenstvom dobre vyrovnala.

Výsledky pre triedu 1 sú v oboch prípadoch na podobnej úrovni. Dokonca, do nesprávnej triedy zasahuje v našom prípade o jednu vzorku menej. Keď však vezmeme do úvahy aj pravdepodobnosť zaradenia vzorky, porovnávaný graf jasne dominuje.



Obrázok 32



Obrázok 33

8 Možnosti budúceho vývoja

Jednou z možností vylepšenia je použiť postup použitý v práci[3], alebo podobný postup, ktorý by z trénovacej množiny vyradil tie prediktory, ktoré zhoršujú výslednú klasifikáciu. Existuje viacero rôznych druhov zhlukovej analýzy ktoré by mohli mať na výsledok pozitívny vplyv.

Možnosťou je takisto pomocou metódy Random Forests predspracovať dáta a určiť dôležité prediktory a takto upravené údaje poskytnúť inej metóde, napríklad SVM.

Ďalšou možnosťou je použitie výsledkov hodnotenia klasifikácie pomocou AUC, alebo priebehu ROC krivky, pre optimalizáciu parametrov trénovania. Analytickejší prístup v tomto smere by určite priniesol kvalitné výsledky, najmä pre metódu SVM. K tomu je priamo v balíčku e1071 k dispozícií nástroj tune.

Čo sa týka samotnej metódy a balíčka randomForest, je možnosť vylepšenia výsledkov pomocou iných balíčkov implementujúcich túto metódu s rôznymi vylepšeniami. Jedným takým je balíček party[19], ktorý navyše poskytuje možnosť generovať podmienené stromy na základe meraní reakcií premenných. Ďalším je napríklad balíček ipred[18], ktorý umožňuje už zmienenú kombináciu s inými modelmi. Rozšírením metódy je metóda randomSurvivalForest[20] ktorá dokáže spracovávať aj prípady kedy niektoré hodnoty premenných nie sú prístupné.

Záver

Pre analýzu dát získaných z DNA čipov je potrebné poznať ich podstatu. Preto je na začiatku práce priblížená táto technológia, jej podstata a princíp. Uvedené sú tiež možné riziká poškodenia dát, ktoré sprevádzajú proces získavania dát z čipov.

Ďalej sú priblížené niektoré tradičné metódy strojového učenia používané pre analýzu dát z DNA čipov. Hlavný dôraz je ale kladený na priblíženie metódy Random Forests. Tieto metódy sú v práci použité k analýze dát génových expresií z nádorov prsníka sledovanej skupiny. Popísané sú vlastnosti tohto datasetu a tiež možnosti ich analýzy. Celá realizácia je vykonaná v prostredí jazyka R. Pre prácu s použitými balíčkami je potrebné pochopiť fungovanie jazyka R a niektoré jeho odlišnosti v porovnaní s inými jazykmi. Popísané sú základné funkcie použitého balíčka ktoré som pri realizácií systému použil. Na datasete som testoval rôzne nastavenia metódy a pokúšal sa dosiahnuť čo najlepší výsledok. Dosiahnuté výsledky som vyhodnotil a porovnal s výsledkami v práci ktorá popisuje analýzu toho istého datasetu ktorý som použil ja, ale pomocou inej metódy. Nakoniec sú spísané niektoré možnosti budúceho vývoja s inými dostupnými balíčkami pre prostredie jazyka R..

Aj napriek tomu že metóda Random Forest v porovnaní neobstála najlepšie, ukázala sa byť účinnou a s širokými možnosťou ďalšieho vylepšenia, najmä vďaka rôznym vedľajším analýzam ktoré poskytuje. Netradičný prístup s určitou mierou náhodnosti ktorý využíva, v sebe skrýva možnosti, ktoré iné metódy nemajú.

Literatúra

[1] Breiman,L., & Cutler,A. (2004), *http*://www.*stat.berkeley.edu/ breiman*/*RandomForests*/

[2] Klaschka J., Kotrč E. (2004) Klasifikační a regresní lesy, sborník konference ROBUST 2004.

[3] Gene expression predictors of breast cancer outcomes
The Lancet, Volume 361, Issue 9369, Pages 1590-1596
E.Huang, S.Cheng, H.Dressman, J.Pittman, M.Tsou, C.Horng, A.Bild, E.Iversen, M.Liao, C.Chen

[4] Biologie, Neil A. Campbell, Jane B. *Reece*, ISBN 8025111784

[5] An Introduction to Bioinformatics Algorithms, Neil C. Jones and Pavel A. Pevzner, ISBN 13 978-0-262-10106-6

[6] M. Schena; D. Shalon; R. W. Davis; P. O. Brown Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science 270, 467 - 470, 1995.

[7] Gojová L., Kozák L. (2006) Možnosti využití DNA čipů v molekulární diagnostice dědičných onemocnění

[8] Gerhold D, Rushmore T, Caskey CT. DNA chips: promising toys have become powerful tools. Trends Biochem 1999;24:168-173.

[9] http://plantandsoil.unl.edu/croptechnology2005/UserFiles/Image/wdyer/

[10] Krontorad, P. Analýza obrazových dat získaných Microarrays skenerem. Brno, 2005. Diplomová práca, Masarykova univerzita.

[11] Machová Kristína: Strojové učenie. Princípy a algoritmy. ELFA s.r.o., 2002, Košice, 117s., ISBN 80 89066 51 8

[12] Berka P. Dobývaní znalostí z databází. Academia 2003, Praha

[13] Zbynek Bortlícek, ROC Krivky, Diplomová práce

[14] http://www.r-project.org/

[15] http://cran.r-project.org/web/packages/randomForest/index.html

[16] http://cran.r-project.org/web/packages/ROCR/index.html

[17] A. Liaw and M. Wiener. Classification and regression by randomForest. R News, 2/3:18-22, December 2002. URL http://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf

[18] http://cran.r-project.org/web/packages/ipred/

[19] http://cran.r-project.org/web/packages/party/index.html

[20] http://cran.r-project.org/web/packages/randomSurvivalForest/index.html

[20] Žižka, J. *Support vector machines*, Studijní material, FI MU Brno, 2006 Dokument http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf?fakulta=1433;obdobi=3523;kod=PA034

[22] Fletcher, T.: Support Vector Machines Explained. 2009. URL www.tristanfletcher.co.uk/SVM%20Explained.pdf

[23] Myslík, V.: Referát z predmetu speciální architektury. http://aldebaran.feld.cvut.cz/~xmyslik/www/neural.htm

[24] Zendulka, J., Bartík, V., Lukáš, R., Rudolfová, I.: Získávání znalostí z databází, Studijní opora. Dostupné na URL: http://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/texts/ZZN.pdf