VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

SROVNÁNÍ HEURISTICKÝCH A KONVENČNÍCH STATISTICKÝCH METOD V DATA MININGU
COMPARISON OF HEURISTIC AND CONVENTIONAL STATISTICAL METHODS IN DATA MINING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Matúš Bitara**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Josef Bednář, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Student:	**Bc. Matúš Bitara**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**Ing. Josef Bednář, Ph.D.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Srovnání heuristických a konvenčních statistických metod v data miningu

Stručná charakteristika problematiky úkolu:

V dnešní době již není problém shromažďovat data jakéhokoli typu, ale je problém získat rozumné informace obsažené v nich. Tato práce se zabývá srovnáním heuristických a konvenčních statistických metod při vytěžování dat včetně stanovení vhodné metody pro srovnání.

Cíle diplomové práce:

Úvod do problematiky heuristických metod.
Matematický popis vhodné konvenční statistické metody.
Matematický popis vhodné heuristické metody.
Aplikace těchto metod na vhodná rozsáhlá data.
Stanovení vhodného postupu pro hodnocení přesnosti metody.
Srovnání heuristických a konvenčních statistických metod.

Seznam doporučené literatury:

HASTIE, Trevor, TIBSHIRANI, Robert a FRIEDMAN, Jerome H. The elements of statistical learning: data mining, inference, and prediction. Corrected ed. New York: Springer, 2003. ISBN 0-387-95284-5.

ANDĚL, Jiří. Základy matematické statistiky. Vyd. 3. Praha: Matfyzpress, 2011. ISBN 978-80-73-8-162-0.

ZVÁRA, Karel. Regresní analýza. Praha: Academia, 1989. ISBN 80-200-0125-5.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Táto práca sa zaoberá porovnaním konvenčných a heuristických metód v data miningu používaných na binárnu klasifikáciu. V teoretickej časti sú popísané štyri rôzne modely. Klasifikácia modelov je demonštrovaná na jednoduchých príkladoch. V praktickej časti sú modely porovnané na reálnych dátach. Táto časť obsahuje aj čistenie dát, odstránenie odľahlých hodnôt, dve rôzne transformácie a redukciu dimenzie. V poslednej časti sú popísané metódy používané na testovanie kvality modelu.

KĽÚČOVÉ SLOVÁ

data mining, logistická regresia, rozhodovacie stromy, náhodný les, gradient boosting, ROC, AUC, Python

ABSTRACT

The thesis deals with the comparison of conventional and heuristic methods in data mining used for binary classification. In the theoretical part, four different models are described. Model classification is demonstrated on simple examples. In the practical part, models are compared on real data. This part also consists of data cleaning, outliers removal, two different transformations and dimension reduction. In the last part methods used to quality testing of models are described.

KEYWORDS

data mining, logistic regression, decision trees, random forest, gradient boosting, ROC, AUC, Python

BIBLIOGRAFICKÁ CITÁCIA

BITARA, M. Srovnání heuristických a konvenčních statistických metod v data miningu. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. 65 s. Vedoucí diplomové práce Ing. Josef Bednář, Ph.D.

ČESTNÉ PREHLÁSENIE

Prehlasujem, že táto práca je mojim pôvodným dielom, spracoval som ju samostatne pod vedením Ing. Josefa Bednáře PhD. a s použitím literatúry uvedenej v zozname.

V Brne dňa 20. mája 2019 ……..……..……..

Matúš Bitara

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Josefovi Bednářovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Taktiež chcem poďakovať svojej rodine, ktorá ma v priebehu štúdia veľmi podporovala.

OBSAH

ÚVOD

Táto práca sa zaoberá porovnaním konvenčných a heuristických metód v data miningu používaných na binárnu klasifikáciu. Cieľom práce je porovnať tieto dva prístupy a na reálnych dátach overiť poznatky získané v teoretickej časti práce.

Prvá kapitola sa zaoberá modelom logistickej regresie. Ide o konvenčný model často používaný v bankovníctve a priemysle. Obľúbený je hlavne vďaka svojej jednoduchej interpretácií. Kapitola stručne popisuje model, spôsob akým sa učí (fituje), a taktiež ako predísť vytvoreniu príliš zložitého modelu. Model je následne otestovaný na jednoduchej úlohe, kde jasne vidieť oblasti klasifikácie. Ďalej sú spomenuté nevýhody modelu a na jednoduchom príklade sú tieto nevýhody graficky demonštrované.

Druhá kapitola sa zaoberá modelom rozhodovacích stromov. Ide o heuristický model s jednoduchou grafickou interpretáciou. V tejto kapitole je popísané jeho fungovanie a taktiež sú popísané parametre modelu, ktoré treba nastaviť ešte pred trénovaním. Kapitola obsahuje dva rovnaké príklady ako v prípade logistickej regresie, na ktorých demonštruje výhody a nevýhody použitia tohto modelu.

Tretia kapitola sa zaoberá skupinovými modelmi, pričom vysvetľuje výhody ich použitia. Ako reprezentant zo skupinových modelov je vybraný náhodný les, ktorý je detailnejšie popísaný. Ide o heuristický model, ktorý je veľmi často používaný v rôznych odvetviach. Patrí medzi najkvalitnejšie modely súčasnosti aj napriek nemožnosti interpretácie rozhodovania tohto modelu. Výhoda použitia tohto modelu je demonštrovaná graficky na jednoduchom príklade.

Štvrtá kapitola sa zaoberá boostingom. Ide o heuristický postup, v ktorom sa vytvorí sekvenčne viac jednoduchých modelov. Modely z tejto kategórie dosahujú všeobecne v mnohých úlohách najlepšie výsledky. Ide o relatívne nové modely. Ako reprezentant tejto skupiny je vybraný XGB (Extreme-Gradient-Boosting) model, ktorý je následne stručne popísaný.

V praktickej časti tejto práce je porovnaná kvalita modelov na reálnych dátach. V tejto časti je taktiež popísané čistenie dát, kontrola odľahlých hodnôt a rôzne transformácie. Ďalej sa tu nachádza redukcia dimenzie dát pomocou jedného z algoritmov popísanom v teoretickej časti. Každý z modelov je otestovaný a výsledky sú v každom kroku porovnané. V poslednej časti kapitoly sú popísané metódy, ktorými je možné vyhodnotiť kvalitu modelov na základe charakteristických grafov.

1 BINÁRNA LOGISTICKÁ REGRESIA

V tejto kapitole sa budem zaoberať jedným z najznámejších modelov matematickej štatistiky a to logistickou regresiou. Logistická regresia nachádza svoje uplatnenie v rôznych odvetviach, od priemyslu, až po bankovníctvo. Veľkou výhodou logistickej regresie v porovnaní s heuristickými metódami je jej jednoduchá interpretácia, pretože sa jedná o takzvaný „*white-box*“ model. Táto vlastnosť zaručuje jej použitie v prípadoch, kde nie je možné používať heuristické modely („*black-box*“). Teóriu použitú v tejto kapitole som čerpal z [1], [2] a [3].

Binárna logistická regresia modeluje pravdepodobnosť pre *Y* pomocou nezávislých náhodných veličín $X_1,\ldots,X_m$. Taktiež predpokladáme, že závislá náhodná veličina je dichotomická, to znamená, že môže nadobúdať len dve hodnoty. Zvyčajne sa jedná o hodnotu 1 pre pozitívnu triedu a 0 pre negatívnu triedu. Nech náhodná veličina *Y* má alternatívne rozdelenie $Y \sim A(\vartheta)$, $0 < \vartheta < 1$. Z toho vyplýva že

$$P(Y = y) = \begin{cases} \vartheta, & y = 1 \\ 1 - \vartheta, & y = 0 \\ 0, & inak. \end{cases}$$

Túto pravdepodobnosť je možne prepísať do jednoduchšej podoby v tvare

$$P(Y = y) = \vartheta^y (1 - \vartheta)^y \quad pre\ y = 0{,}1.$$

Stredná hodnota a rozptyl alternatívneho rozdelenia nadobúdajú hodnoty

$$EY = \vartheta \qquad DY = \vartheta(1 - \vartheta).$$

Pretože *Y* je dichotomická veličina, tak platí $P(Y = 1) = 1 - P(Y = 0)$. Z toho je zrejmé, že si môžeme vybrať, či modelovať $P(Y = 1)$ alebo $P(Y = 0)$. V praxi sa však častejšie používa prípad $P(Y = 1)$ a tak ho budem požívať aj ja v tejto práci. Táto pravdepodobnosť nadobúda hodnoty z intervalu $\langle 0{,}1\rangle$. Mohlo by sa zdať, že vytvorený model $P(Y = 1) = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m$ bude vhodný na modelovanie pravdepodobnosti *Y*. Toto tvrdenie však nie je pravda, pretože určité realizácie $x_1, \ldots, x_m$ veličín $X_1, \ldots X_m$ môžu lineárnou kombináciou s koeficientami $\beta_0, \ldots \beta_m$ nadobúdať hodnoty mimo intervalu $\langle 0{,}1\rangle$.

1.1 ŠANCA (ANGLICKY ODDS)

Tento problém je možné čiastočne odstrániť zavedením pojmu šanca.

$$odds\big(P(Y = 1)\big) = \frac{P(Y = 1)}{P(Y = 0)} = \frac{P(Y = 1)}{1 - P(Y = 1)}$$

Šanca vyjadruje, koľkokrát je väčšia pravdepodobnosť, že *Y* nadobudne hodnotu 1, než pravdepodobnosť, že *Y* nadobudne hodnotu 0. Hodnoty šance ležia v intervale $(0, \infty)$. Teraz je ešte nutné jednoznačne transformovať interval $(0, \infty)$ na interval $(-\infty, \infty)$. Na tento účel sa použije prirodzený logaritmus, pomocou ktorého je zavedená logitová funkcia.

1.2 LOGITOVÁ FUNKCIA

Logitovú funkciu

$$logit\big(P(Y=1)\big)$$

vyjadríme v tvare

$$\begin{aligned}\text{logit}\big(\text{P}(\text{Y}=1)\big) &= \ln\Big(\text{odds}\big(\text{P}(\text{y}=1)\big)\Big)\\ &= \ln\left(\frac{\text{P}(\text{Y}=1)}{1-\text{P}(\text{Y}=1)}\right).\end{aligned}$$

Takto transformovaná pravdepodobnosť už nadobúda hodnoty z intervalu $(-\infty, \infty)$ a môžeme ju modelovať podobne ako je to u lineárnej regresie

$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m.$$

Z tejto rovnice vyjadríme pravdepodobnosť $P(Y=1)$ nasledovne

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots+\beta_m X_m)}}.$$

Pri označení $\boldsymbol{\beta} = (\beta_0, \ldots \beta_m)^T$ a $\boldsymbol{X} = (1, X_1, \ldots X_m)^T$ môžeme tento vzťah prepísať do tvaru

$$P(Y=1) = \frac{1}{1+e^{-\boldsymbol{X}^T\boldsymbol{\beta}}}.$$

1.3 MODEL

Pretože pre rôzne realizácie $\boldsymbol{x}$ náhodného vektora $\boldsymbol{X}$ nadobúda pravdepodobnosť rôznych hodnôt, táto pravdepodobnosť je podmienená a môžeme ju vyjadriť ako

$$P(Y=1|\boldsymbol{X}=\boldsymbol{x}) = \frac{1}{1+e^{-x^T\boldsymbol{\beta}}}. \qquad (1)$$

Pre úplnosť môžeme odvodiť model doplnkovej pravdepodobnosti ako

$$P(Y=0|\boldsymbol{X}=\boldsymbol{x}) = 1 - P(Y=1|\boldsymbol{X}=\boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{x}^T\boldsymbol{\beta}}}.$$

Môžeme si všimnúť, že $P(Y=1) = \vartheta = \text{EY}$, to znamená, že modelom predikujeme strednú hodnotu náhodnej veličiny Y v závislosti na realizáciách $\boldsymbol{x}$.

K vytvoreniu logistického regresného modelu je potrebné mať dostatočný počet meraní. Majme súbor o rozsahu n. Nech $Y_1,\ldots,Y_n \sim A(\vartheta)$, $0 < \vartheta < 1$. Označme y_i realizáciu náhodnej veličiny Y_i. Nech ďalej ku každej Y_i pripadá súbor m náhodných veličín $X_{i1}, \ldots, X_{im}$ s realizáciami

$x_{i1}, \dots, x_{im}$. Označme $\boldsymbol{x} = (x_{i1}, \dots, x_{im})^T$. Potom model logistickej regresie pre *i*-te pozorovanie je v tvare

$$P(Y = 1|\boldsymbol{X_i} = \boldsymbol{x_i}) = \frac{1}{1 + e^{-\boldsymbol{x_i}^T\boldsymbol{\beta}}}.$$

1.4 POMER ŠANCÍ

Z (1) vyplýva, že pokiaľ je nejaké $\beta_i > 0$ a x_i porastie, tak porastie aj pravdepodobnosť $P(Y = 1)$. Čím väčšie bude β_i, tým bude tento nárast rýchlejší. Zavedieme pojem pomer šancí (anglicky odds ratio) nasledujúcim vzťahom

$$OR(X_i) = \frac{odds(P(Y = 1|X_1 = x_1, \dots, X_{i-1} = x_{i-1}, X_i = x_i + 1, X_{i+1} = x_{i+1}, \dots, X_m = x_m))}{odds(P(Y = 1|X_1 = x_1, \dots, X_{i-1} = x_{i-1}, X_i = x_i, X_{i+1} = x_{i+1}, \dots, X_m = x_m))}.$$

Po dosadení z (1) dostaneme

$$OR(X_i) = e^{\beta_i}.$$

Pomer šancí $OR(X_i)$ udáva, koľkokrát sa zväčší šanca na to, aby sa Y realizovala hodnotou 1 keď sa hodnota nezávislej premennej zvýši o 1, pričom ostatné nezávislé premenné ostanú fixované. Tento pomer je jednoznačne určený koeficientom β_i. V prípade že je $\beta_i > 0$ je $OR(X_i) > 1$. V prípade že je $\beta_i < 0$ je $OR(X_i) < 1$.

1.5 VYTVORENIE ÚČELOVEJ FUNKCIE (LOSS FUNCTION)

V predchádzajúcich kapitolách bolo uvedené ako model logistickej regresie počíta pravdepodobnosť pozitívnej triedy. Ostáva získať odhad vektora parametrov $\boldsymbol{\beta}$. Cieľom je nájsť taký odhad vektora parametrov $\boldsymbol{\beta}$, s ktorým model odhadne vysokú pravdepodobnosť pozitívnej triedy pre pozitívne pozorovania ($y = 1$) a nízku pravdepodobnosť pozitívnej triedy pre negatívne pozorovania ($y = 0$). Za týmto účelom je nutné zaviesť účelovú funkciu, ktorej maximalizáciou získame optimálny odhad vektora parametrov $\boldsymbol{\beta}$. Účelová funkcia pre logistickú regresiu je v tvare

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(Y_i = y_i) = \prod_{i=1}^{n} \frac{(e^{-\boldsymbol{x_i}^T\boldsymbol{\beta}})^{1-y_i}}{1 + e^{-\boldsymbol{x_i}^T\boldsymbol{\beta}}}$$

Táto funkcia sa nazýva tiež ako vierohodnostná funkcia a je odvodená metódou maximálnej vierohodnosti, ktorá je popísaná v [4]. Pre nás je však výhodnejšie pracovať s logaritmickou verziou tejto funkcie. Po niekoľkých úpravách dostávame

$$l(\boldsymbol{\beta}) = \ln\big(L(\boldsymbol{\beta})\big) = \frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln\left(\frac{1}{1 + e^{-\boldsymbol{x_i}^T\boldsymbol{\beta}}}\right) + (1 - y_i)\ln\left(1 - \frac{1}{1 + e^{-\boldsymbol{x_i}^T\boldsymbol{\beta}}}\right)\right].$$

Vynásobením hodnotou -1 dostávame z účelovej funkcie, ktorú chceme maximalizovať, účelovú funkciu, ktorú chceme minimalizovať v tvare

$$J(\boldsymbol{\beta}) = -\ln\big(L(\boldsymbol{\beta})\big) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln\left(\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right) + (1-y_i)\ln\left(1-\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right)\right].$$

Táto funkcia je priemer cez všetky pozorovania z dát o rozsahu n na ktorých model trénujeme. Princíp funkcie je jednoduché vysvetliť na jednom pozorovaní. V prípade, že pozorovanie je pozitívnej triedy ($y = 1$), člen

$$(1-y_i)\ln\left(1-\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right)$$

je rovný nule. Ak teda model predikuje vysokú pravdepodobnosť pozitívnej triedy, logaritmus tejto pravdepodobnosti je blízky nule a celá funkcia nadobúda hodnotu blízku nule. V prípade, že pozorovanie je negatívnej triedy ($y = 0$), člen

$$y_i \ln\left(\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right)$$

je rovný nule. Predikcia s vysokou pravdepodobnosťou pozitívnej triedy je teda penalizovaná logaritmom

$$\ln\left(1-\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right)$$

ktorý nadobúda hodnoty idúce do nekonečna pre argument v limite blízky nule.

Na odhad vektora $\boldsymbol{\beta}$ však neexistuje ekvivalent normálnej rovnice v tvare

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{y})$$

ako je tomu pri lineárnej regresií. Dobrá správa však je, že funkcia je konvexná a teda metódou Gradient Descent je možné nájsť globálne minimum účelovej funkcie (v prípade že rýchlosť učenia nie je príliš vysoká a počkáme dostatočne dlho).

1.6 GRADIENT DESCENT

Gradient Descent je optimalizačná metóda, ktorá sa používa na hľadanie minima účelovej funkcie. V našom prípade hľadáme taký odhad vektora parametrov $\boldsymbol{\beta}$, s ktorým účelová funkcia $J(\boldsymbol{\beta})$ nadobúda globálne minimum.

Algoritmus je nasledovný:

1. Spočítaj gradient funkcie $J(\boldsymbol{\beta})$ ako

$$grad\, J(\boldsymbol{\beta}) = \nabla J = \left(\frac{\partial J}{\partial \beta_1}, \dots, \frac{\partial J}{\partial \beta_m},\right)$$

2. Inicializuj náhodne vektor parametrov $\boldsymbol{\beta}$
3. Dosaď náhodne inicializované hodnoty vektora $\boldsymbol{\beta}$ do gradientu
4. Spočítaj krok ako

$$\boldsymbol{step} = \eta \nabla J$$

kde η je rýchlosť učenia (anglicky learning rate). Rýchlosť učenia sa obvykle volí z intervalu $(0,1)$.

5. Spočítaj nový vektor parametrov $\boldsymbol{\beta}$ ako

$$\boldsymbol{\beta} := \boldsymbol{\beta} - step.$$

6. Opakuj kroky 3 – 5, dokiaľ hodnota kroku nie je menšia ako zvolená hodnota (obvykle sa ako stop kritérium pre veľkosť kroku používa hodnota 0,001), alebo dokiaľ algoritmus nepresiahne zvolený počet iterácií (často sa používa hodnota 1000)

Ako nevýhodu tohto algoritmu môžeme považovať fakt, že v každej iterácii používa na výpočet gradientu všetky pozorovania z trénovacej množiny. V prípade pár tisícov pozorovaní to problém nie je, ale v prípade trénovacej množiny s miliónmi pozorovaní to už problém môže byť. Ako alternatívu je možné použiť Stochastic gradient descent, ktorý v každej iterácii počíta gradient iba z jedného náhodne vybraného pozorovania z trénovacej množiny. Porovnanie týchto algoritmov je detailnejšie popísane v [5].

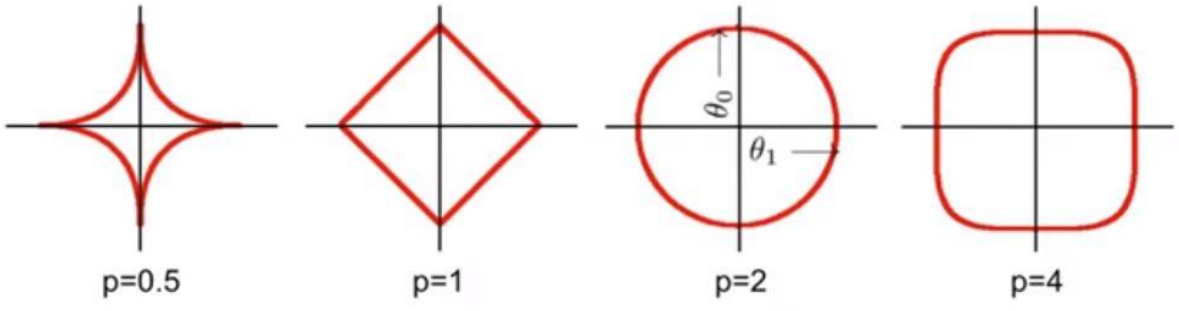
1.7 REGULARIZÁCIA MODELU

Pretrénovanie modelu (anglicky overfitting) je klasický problém pre mnohé algoritmy. Tento problém nastáva, keď predikcie modelu sú výrazne presnejšie na trénovacej množine ako na testovacej. Hovoríme, že model má vysoký rozptyl (anglicky high variance), čo môže byť spôsobené veľkým počtom parametrov v modeli. Opačný prípad je podtrénovanie modelu (anglicky underfitting), ktoré sa vyznačuje vysokým skreslením modelu (anglicky high bias). V tomto prípade je model príliš jednoduchý a nepopisuje dáta dostatočne kvalitne.

Keďže nechceme, aby predikcie modelu boli presné iba na trénovacej množine, ale aj na dátach ktoré model nikdy nevidel, musíme zaviesť regularizačný člen nasledovne

$$L_p(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_p = \lambda(\sum_{i=1}^{m} |\beta_i|^p)^{\frac{1}{p}}$$

kde parametrom λ určujeme silu regularizácie (čím väčšia hodnota λ, tým silnejšia regularizácia). Pre $m = 2$ a $\|\boldsymbol{\beta}\|_p = c$, kde c je ľubovoľná kladná konštanta, môžeme vykresliť regularizačný člen pre rôzne hodnoty p.



Obrázok 1: Vplyv regularizačného parametra p (prevzaté z [5])

Z obrázka je možne vidieť, že pre $p \geq 1$ je regularizačný člen konvexný a tým pádom ľahko optimalizovateľný. Podrobnejšie je regularizačný člen vysvetlený v [5]. V tejto práci budem používať len L_1 a L_2 regularizácie. L_1, L_2 regularizácie definujeme nasledovne

$$L_1(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\| = \lambda \sum_{i=1}^{m} |\beta_i|$$

$$L_2(\boldsymbol{\beta}) = \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2 = \frac{\lambda}{2}\sum_{i=1}^{m} \beta_i^2$$

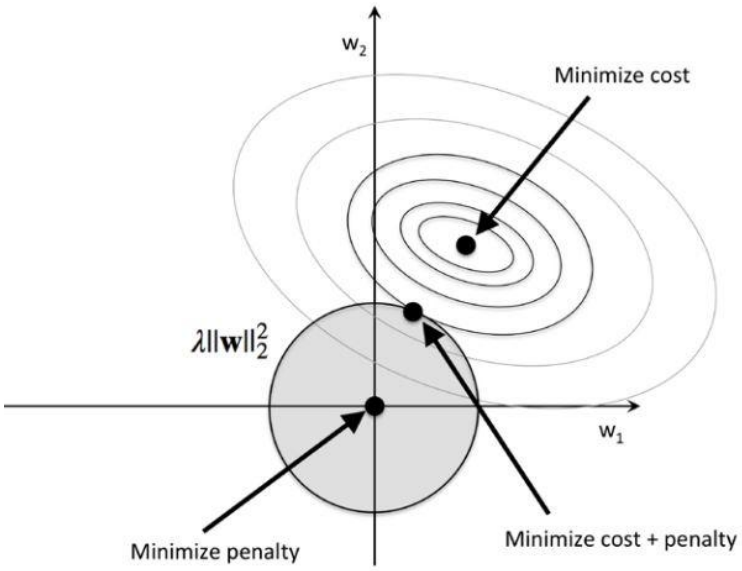
Pridaním $L_2(\boldsymbol{\beta})$ regularizácie do účelovej funkcie dostávame

$$J(\boldsymbol{\beta}) = \ln\big(L(\boldsymbol{\beta})\big)$$
$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln\left(\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right) + (1-y_i)\ln\left(1-\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right)\right] + \frac{\lambda}{2}\sum_{i=1}^{m}\beta_i^2$$

obdobne môžeme funkciu upraviť pre $L_1(\boldsymbol{\beta})$ regularizáciu

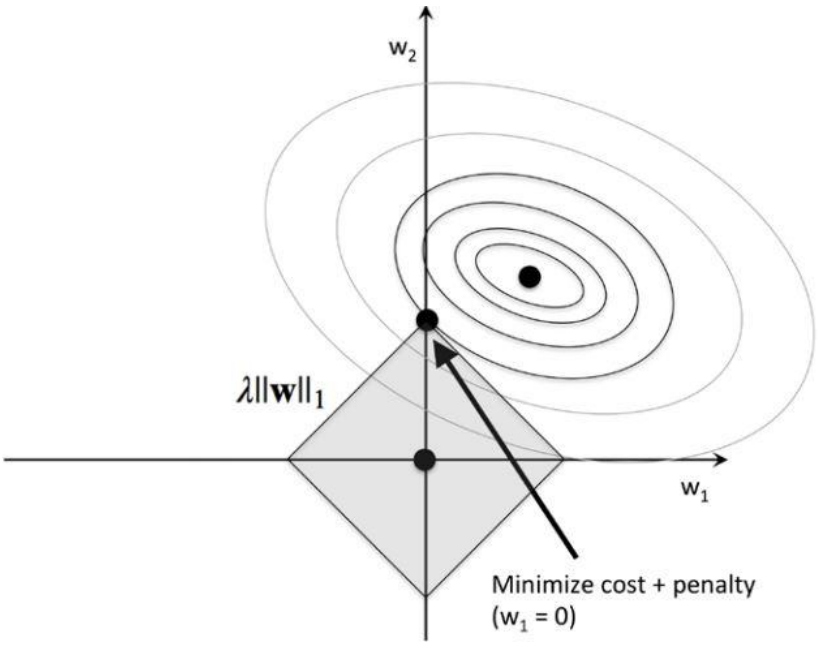
$$J(\boldsymbol{\beta}) = \ln\big(L(\boldsymbol{\beta})\big)$$
$$= -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln\left(\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right) + (1-y_i)\ln\left(1-\frac{1}{1+e^{-x_i{}^T\boldsymbol{\beta}}}\right)\right] + \lambda\sum_{i=1}^{m}|\beta_i|$$

Pre lepšie znázornenie fungovania L_1 a L_2 regularizácií je vhodné tento princíp ilustrovať na konvexnej funkcií v 2D. Obrázok 2 znázorňuje konvexnú funkciu s dvoma parametrami w_1, w_2. Šedý kruh znázorňuje L_2 regularizačný člen. Cieľom je zvoliť tieto parametre tak, aby hodnota účelovej funkcie bola čo najnižšia a zároveň, aby parametre spĺňali podmienky regularizácie, to jest aby patrili do šedého kruhu. Polomer tohto kruhu určuje parameter λ. Čím má λ väčšiu hodnotu, tým je polomer kruhu menší. Hľadáme teda taký bod, kde kruh pretína vrstevnice účelovej funkcie a v ktorom je zároveň hodnota účelovej funkcie najmenšia. V prípade voľby príliš vysokej hodnoty λ sa z kruhu stane len bod v počiatku súradníc a celý model sa znehodnotí.



Obrázok 2: L_2 regularizácia (prevzaté z [11])

Myšlienka L_1 regularizácie je veľmi podobná ako pri L_2. Keďže L_1 regularizácia používa súčet absolútnych hodnôt koeficientov namiesto súčtu kvadrátov koeficientov ako je to pri L_2, výsledná oblasť regularizácie teda nie je kruh, ale diamantový tvar. L_1 regularizácia je znázornená na obrázku 3. Z obrázka je možné vidieť, že vrstevnice účelovej funkcie sa dotýkajú oblasti regularizácie v bode so súradnicou $w_1 = 0$. Pri tomto type regularizácie je veľmi pravdepodobné, že optimálny bod bude mať jednu, alebo viac súradníc nulových. Táto vlastnosť sa dá využiť v prípade, že máme dáta s veľkým počtom nezávislých premenných, z ktorých niektoré nie sú relevantné. L_1 regularizácia vynuluje koeficienty pri nevýznamných premenných a zníži dimenziu dát. Preto môžeme túto regularizáciu považovať aj ako techniku na zníženie dimenzie dát.



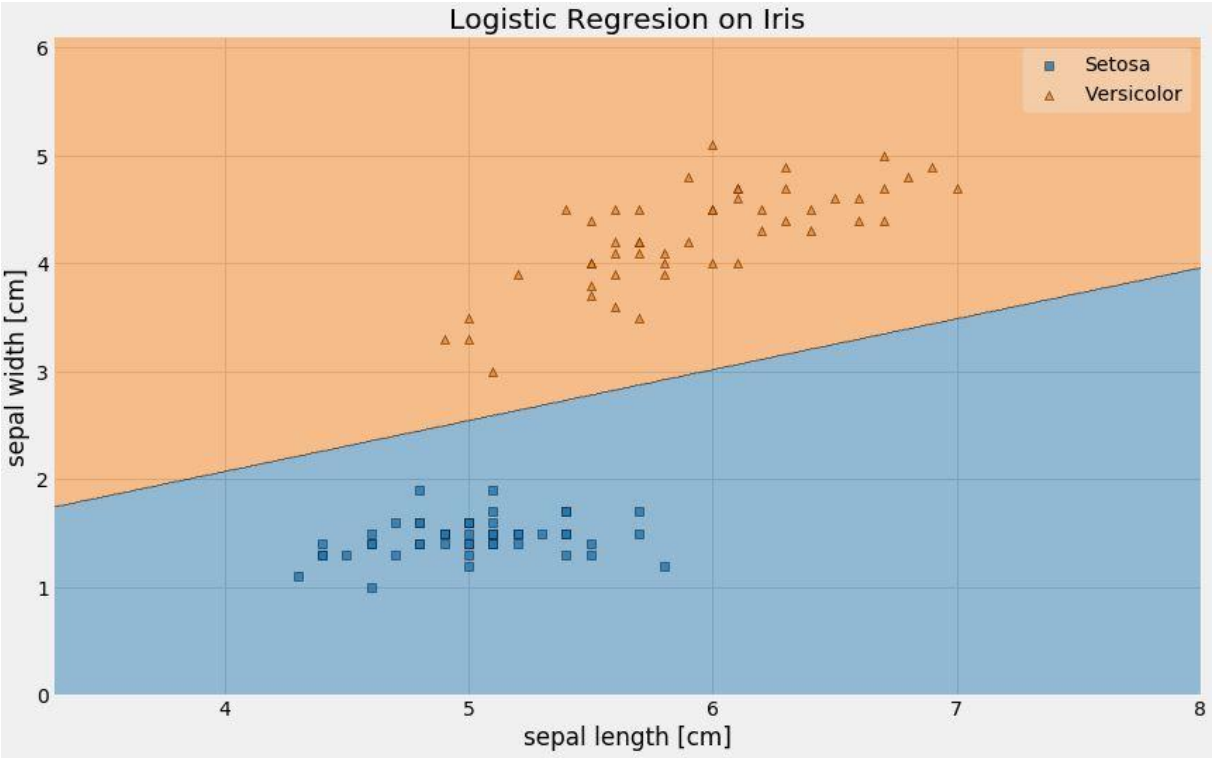
Obrázok 3: L_1 regularizácia (prevzaté z [11])

1.8 ILUSTRAČNÉ PRÍKLADY

V tejto kapitole uvediem dva príklady, na ktorých budem ilustrovať fungovanie viacerých modelov v tejto práci. V prvom príklade pôjde o lineárne separovateľné dáta a v druhom naopak o nelineárne separovateľné dáta. Obidva príklady používajú pre jednoduchosť a možnosť vykreslenia len dva prediktory a jednu závislú premennú. Príklady sú vytvorené s využitím programovacieho jazyka Python [16]. Zdrojový kód ilustračných príkladov je možné nájsť v prílohe na CD.

1.8.1 LINEÁRNE SEPAROVATEĽNÉ DÁTA

V tomto prípade ako dáta použijem zjednodušenú verziu Iris datasetu, ktorý bol predstavený štatistikom a biológom Ronaldom Fischerom v roku 1936 v jeho článku [6]. Dataset je voľne dostupný z [7]. Originálny dataset obsahuje 150 pozorovaní troch druhov kosatca (po anglicky iris). Každý druh sa v datasete nachádza 50 krát. Každé pozorovanie obsahuje informáciu o dĺžke a šírke kališného lístka, dĺžke a šírke okvetného lístka a druhu kosatca. Dataset s ktorým budem pracovať, však obsahuje iba šírku kališného lístka a dĺžku kališného lístka. Taktiež som vybral iba dva druhy kosatca a to odrody Setosa a Versicolor. Cieľom tohto príkladu je ukázať ako logistická regresia rozdelí priestor prediktorov na dve časti. Toto rozdelenie znázorňuje obrázok 4.



Obrázok 4: Logistická regresia - lineárne separovateľné dáta

Body v tvare štvorca znázorňujú druh Iris Setosa a body v tvare trojuholníka znázorňujú Iris Versicolor. Hranica rozdeľuje priestor prediktorov na dve oblasti. Pozorovania spadajúce do modrej oblasti budú logistickou regresiou klasifikované ako druh Iris Setosa a pozorovania spadajúce do oranžovej oblasti budú klasifikované ako Iris Versicolor. Z obrázka je možné

vidieť, že logistická regresia kategorizovala všetky pozorovania správne a dokázala odlíšiť tieto dva druhy. Koeficienty logistickej regresie boli v tomto prípade nasledovné:

$$\widehat{\beta_0} = -0.55$$

$$\widehat{\beta_1} = -1.41$$

$$\widehat{\beta_2} = 2.98.$$

Z týchto koeficientov je jednoduché vyjadriť priamku znázorňujúcu hranicu z obrázka ako

$$\widehat{\beta_0} + \widehat{\beta_1} * sepal\_length + \widehat{\beta_2} * sepal\_width = 0$$

1.8.2 LINEÁRNE NESEPAROVATEĽNÉ DÁTA

V predchádzajúcom príklade bol model logistickej regresie schopný rozdeliť dané pozorovania správne. Čo sa však stane, ak dáta nebudú lineárne separovateľné? Na vygenerovanie lineárne neseparovateľných dát použijem knižnicu Sklearn [23], v ktorej je implementovaná funkcia make_circles. Táto funkcia vygeneruje dáta zobrazené na obrázku číslo 5.

Obrázok 5: Logistická regresia – lineárne neseparovateľné dáta

Z obrázka je opäť možné vidieť pozorovania dvoch kategórií a hranicu, podľa ktorej logistická regresia rozdeľuje oblasť na dve časti. Keďže pozorovania z kategórie 1 sa nachádzajú aj v časti predikcie kategórie 0 a naopak, model je veľmi nepresný. Možnosť ako zlepšiť tento model je nahradiť model v tvare

$$P(Y = 1|\boldsymbol{X_i} = \boldsymbol{x_i}) = \frac{1}{1 + e^{-{x_i}^T \boldsymbol{\beta}}}$$

modelom v tvare

$$P(Y = 1|\boldsymbol{X_i} = \boldsymbol{x_i}) = \frac{1}{1 + e^{-{x_i}^{2^T} \boldsymbol{\beta}}}.$$

Touto modifikáciou však stratíme jednu z najväčších výhod logistickej regresie a tou je jednoduchá interpretácia. Predstavme si model, ktorý predikuje, či je osoba muž, alebo žena na základe nameranej výšky a váhy. V nemodifikovanom modeli by bola interpretácia koeficientov jednoduchá, avšak pri použití kvadrátu je model len ťažko vysvetliteľný, pretože do neho vstupujú výška a váha v kvadráte, čo nedáva logicky žiadny zmysel.

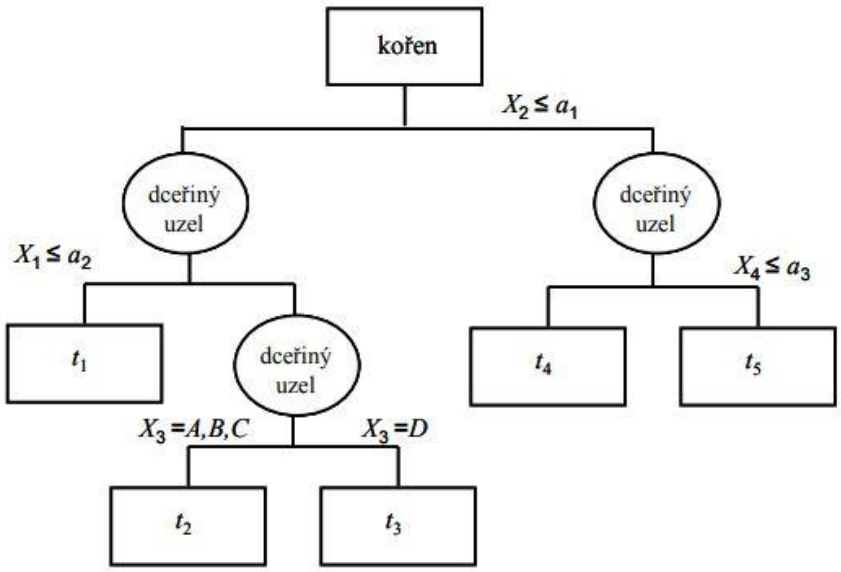
Pretože logistická regresia nedokáže dostatočne dobre predikovať prípady s lineárne neseparovateľnými dátami je nutné pri modelovaní používať aj algoritmy, ktoré tento problém dokážu vyriešiť. Niektoré z nich budú popísané v nasledujúcich kapitolách.

2 ROZHODOVACIE STROMY

Rozhodovací strom tvorí sada hierarchicky usporiadaných rozhodovacích pravidiel. Medzi rozhodovacím stromom a stromom v prírode je tiež určitá analógia. Rozhodovací strom obsahuje listy, koreň, rastie a taktiež ho môžeme prerezávať. Koreň stromu predstavuje celý dátový súbor a postupne prebieha vetvenie do ďalších uzlov. Uzly, ktoré sa už ďalej nedelia nazývame terminálne uzly, ale používa sa aj názov listy. V tomto texte sa budem zaoberať binárnymi stromami, ale existujú aj nebinárne stromy. Viac o nebinárnych stromoch je možné nájsť v [9]. Veľkou výhodou rozhodovacích stromov je, že nemajú žiadne nároky na rozloženie dát ako sú napríklad nezávislosť prediktorov, normálne rozdelenie a iné. V prípade binárnych stromov sa uzly vetvia na dve vetvy, v prípade nebinárnych je vetví viac. Rozhodovacie stromy môžeme rozdeliť podľa typu závislej premennej Y na klasifikačné a regresné. Ďalej sa budem zaoberať iba binárnymi klasifikačnými stromami a to konkrétne algoritmom CART (Classification and Regresion Trees). Teóriu použitú v tejto kapitole som čerpal z [9] a [10].

2.1 CART

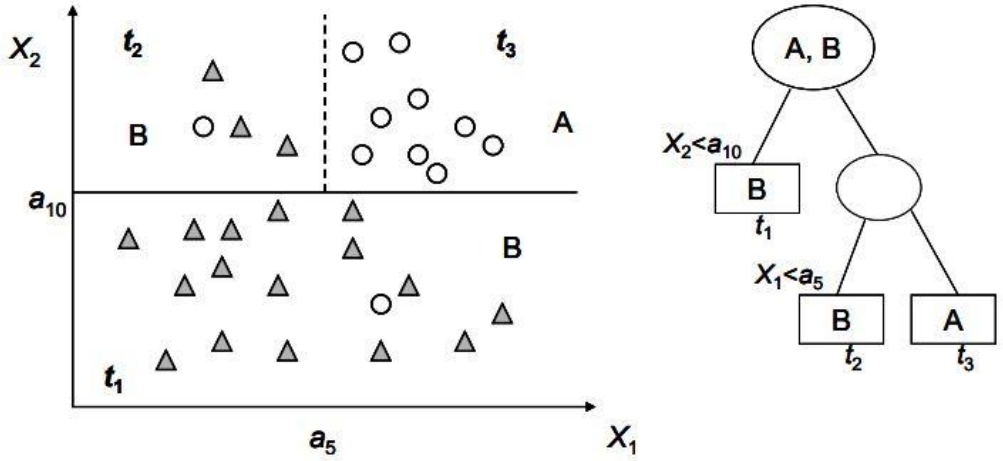
Stromy typu CART rastú na základe binárneho delenia. Na začiatku tvorby stromu patria všetky pozorovania z dát do jedného uzla nazývaného koreň. Následne sú tieto pozorovanie rozdelené do dvoch dcérskych uzlov na základe hodnoty a a prediktoru $X_i, i = 1, \dots, m$. Toto delenie pokračuje až do chvíle, kým sa v každom uzle nachádza len jedno pozorovanie, alebo kým nenastane jedno zo zvolených stop kritérií. Tieto kritériá budú popísané v texte ďalej. Jednoduchý strom je možné vidieť na obrázku číslo 6.



Obrázok 6: Rozhodovací strom (prevzaté z [9])

Indexy u terminálnych uzlov udávajú v akom poradí strom vyrástol. Prediktory X_1, X_2, X_4 sú spojité, X_3 je kategoriálny s kategóriami *A,B,C,D.*

Hodnoty prediktorov použité pri vetvení rozdeľujú daný m-dimenzionálny priestor na oblasti nazývané taktiež aj ako regióny R_t.



Obrázok 7: Rozhodovací strom – oblasti (prevzaté z [9])

Zostáva nám zistiť ako nájsť správne rozdelenie. Úlohou je nájsť také rozdelenie závislej premennej Y prediktorom X_i, aby hodnoty premennej Y boli v uzle čo najviac homogénne a zároveň čo najrozdielnejšie medzi uzlami. Ktorý prediktor X_i a jeho hodnota zaistí najlepšie rozdelenie, zistíme pomocou kriteriálnej štatistiky, ktorá určuje homogenitu uzla. Najpoužívanejšie kriteriálne štatistiky pre binárne klasifikačné stromy sú

$$Gini\ index\colon\ GI_t = 1 - \sum_{c=1}^{2} p_{t,c}^2$$

$$Entropia\colon\ H_t = -\sum_{c=1}^{2} p_{t,c}\, log_2(p_{t,c})$$

$$Klasifikačná\ chyba\colon\ ME_t = 1 - max\{p_{t,c}\}$$

kde $p_{t,c}$ je pravdepodobnosť kategórie c v uzle t. Najpoužívanejšou kriteriálnou štatistikou je *Gini index.* Táto štatistika nadobúda hodnotu rovnú nule v prípade, že uzol je dokonale čistý, to znamená, že všetky pozorovania v uzle sú rovnakej kategórie. Naopak, štatistika nadobúda maximálnu hodnotu 0,5 v prípade, že je v uzle rovnaký počet pozorovaní z každej kategórie. Po rozdelení uzla na dva dcérske uzly dôjde k vypočítaniu *Gini indexu* pre každý dcérsky uzol zvlášť. Celkovú hodnotu *Gini indexu* pre dané rozdelenie označíme ako GI_{tot} a vypočítame ju ako vážený súčet *Gini indexov* dcérskych uzlov. Ako váhy použijeme pomer počtu pozorovaní v danom dcérskom uzle ku počtu pozorovaní v materskom uzle. Matematicky môžeme GI_{tot} vyjadriť nasledovne

$$GI_{tot} = \sum_{i=1}^{2} \frac{n_i}{n_t} \mathrm{GI}_i$$

kde n_i je počet pozorovaní v *i*-tom dcérskom uzle a n_t je počet pozorovaní v materskom uzle s indexom *t*. Obdobne môžeme vypočítať celkovú hodnotu *entropie* H_{tot} ako

$$H_{tot} = \sum_{i=1}^{2} \frac{n_i}{n_t} \mathrm{H}_i.$$

Hodnoty *entropie* nadobúdajú maximum a minimum v rovnakých prípadoch ako to bolo pri *Gini indexe*. Maximum v prípade, že je v uzle rovnaký počet pozorovaní z každej kategórie a minimum pri dokonale čistom uzle.

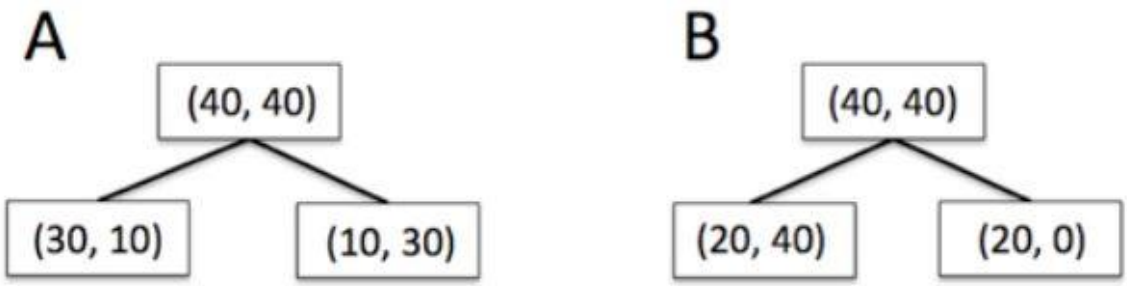
Ďalším kritériom, ktoré môžeme použiť pri rozdelení stromu je *informačný zisk* (*GAIN*), ktorý meria pokles entropie. Informačný zisk vyjadríme nasledovne

$$GAIN_{tot} = \mathrm{H}_t - \left(\sum_{i=1}^{2} \frac{n_i}{n_t} \mathrm{H}_i \right).$$

Posledným kritériom je celková *klasifikačná chyba* ME_{tot}, ktorá sa spočíta podobne ako H_{tot}.

$$ME_{tot} = \sum_{i=1}^{2} \frac{n_i}{n_t} \mathrm{ME}_i$$

Gini index a *entropia* sa obvykle používajú pri raste stromu na rozdiel od *klasifikačnej chyby*, ktorá sa používa pri orezávaní stromu. Keďže *Gini index* a *entropia* majú veľmi podobné výsledky, odporúča sa používať skôr *Gini index* z dôvodu jednoduchšieho výpočtu. Prečo nie je vhodné používať *klasifikačnú chybu* pri raste stromu si ukážeme na nasledujúcom príklade. Predstavme si dve možné rozdelenia uzla, ktoré sú zobrazené na obrázku číslo 8.



Obrázok 8: Delenie uzla

V obidvoch prípadoch máme v materskom uzle 40 pozorovaní z každej kategórie. Začnime výpočtom *Gini indexu* pre obidve rozdelenia.

Rozdelenie A:

$$IG(D_p) = 1 - (0.5)^2 - (0.5)^2 = 0.5$$

$$IG(D_{left}) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$IG(D_{right}) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$IG_{tot} = \frac{4}{8}0.375 + \frac{4}{8}0.375 = 0.375$$

Rozdelenie B:

$$IG(D_p) = 1 - (0.5)^2 - (0.5)^2 = 0.5$$

$$IG(D_{left}) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.44$$

$$IG(D_{right}) = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$$

$$IG_{tot} = \frac{6}{8}0.44 + \frac{2}{8}0 = 0.33$$

Z výsledkov je možné vidieť, že podľa hodnoty IG_{tot} je výhodnejšie rozdelenie B, pretože má nižšiu hodnotu. Obdobne by dopadol výpočet *entropie*. Poďme sa však pozrieť ako dopadne výpočet s použitím *klasifikačnej chyby*.

Rozdelenie A:

$$IE(D_p) = 1 - 0.5 = 0.5$$

$$IE(D_{left}) = 1 - \left(\frac{3}{4}\right) = 0.25$$

$$IE(D_{right}) = 1 - \left(\frac{3}{4}\right) = 0.25$$

$$IE_{tot} = \frac{4}{8}0.25 + \frac{4}{8}0.25 = 0.25$$

Rozdelenie B:

$$IE(D_p) = 1 - (0.5) = 0.5$$

$$IE(D_{left}) = 1 - \left(\frac{4}{6}\right) = \frac{1}{3}$$

$$IE(D_{right}) = 1 - \left(\frac{1}{1}\right) = 0$$

$$IE_{tot} = \frac{6}{8}\frac{1}{3} + \frac{2}{8}0 = 0.25$$

V tomto prípade hodnoty IE_{tot} rozdelenia A aj B sú rovnaké. Dôvodom prečo *klasifikačná chyba* nedokázala rozlíšiť medzi rozdeleniami je citlivosť na zmenu pravdepodobností v uzloch. *Gini index* a *entropia* sú oveľa viac citlivé na zmeny pravdepodobností kategórii v uzloch ako *klasifikačná chyba* a preto sú ako kriteriálne štatistiky vhodnejšie na rast stromu. Všeobecne je doporučené používať skôr *Gini index* ako *entropiu* z dôvodu veľmi podobných výsledkov. Výhoda *Gini indexu* oproti *entropii* je taktiež ušetrenie výpočtového času, pretože výpočet neobsahuje logaritmy.

2.2 PRIRADENIE HODNOTY LISTU

Pri klasifikačnom strome je každému listu priradená hodnota výslednej kategórie závislej premennej. Výslednú kategóriu listu berieme tú, ktorá má v danom liste najväčšie zastúpenie. Nové pozorovanie je tak klasifikované podľa kategórie listu, do ktorého je stromom zaradené. Môže však nastať problém, že obidvom dcérskym listom bude priradená rovnaká kategória. Táto situácia môže nastať, keď sú kategórie nevyvážené. Pre lepšie pochopenie nám poslúži nasledujúci príklad.

Majme 110 pozorovaní, pričom 10 je z pozitívnej kategórie (značenej ako 1) a 100 pozorovaní je z negatívnej (značenej ako 0). V koreni stromu je teda GI_{root} rovný hodnote

$$GI_{root} = 1 - \left(\frac{10}{110}\right)^2 - \left(\frac{100}{110}\right)^2 = 0.165$$

Ďalej si predstavme, že algoritmus rozdelil tento uzol na dva dcérske uzly, označené ako *L* a *R*. V dcérskom uzle *L* sa nachádza 50 pozorovaní z negatívnej triedy a 2 z pozitívnej triedy. V dcérskom uzle *R* sa nachádza 50 pozorovaní z negatívnej triedy a 8 z pozitívnej. Spočítame GI_L, GI_R a GI_{tot}

$$GI_L = 1 - \left(\frac{50}{52}\right)^2 - \left(\frac{2}{52}\right)^2 = 0.0739$$

$$GI_R = 1 - \left(\frac{50}{58}\right)^2 - \left(\frac{8}{58}\right)^2 = 0.23$$

$$GI_{tot} = \frac{52}{110} * 0.0739 + \frac{58}{110} * 0.23 = 0.156$$

Z výpočtov je možné vidieť, že hodnota $GI_{root} > GI_{tot}$ a tým pádom rozdelenie podľa kriteriálnej štatistiky malo význam. Keďže však v uzle *L* je väčšia časť pozorovaní z negatívnej kategórie, výsledná kategória uzla bude negatívna. Podobná situácia nastáva v uzle *R*. Týmto spôsobom sme dostali dva dcérske uzly s rovnakou výslednou kategóriou, čo nedáva zmysel a predikciu stromu nijako nezlepší. V takomto prípade je možné použiť váženie jednotlivých

kategórií na získanie výslednej kategórie uzla. V našom prípade by uzol *L* bol negatívnej výslednej kategórie a uzol *R* pozitívnej výslednej kategórie.

2.3 ALGORITMUS RASTU STROMU

Algoritmus rastu stromu popíšeme nasledovne :

1. Nájdi najlepšie rozdelenie každého z prediktorov X_i:
   - V prípade spojitého prediktora X_i zoraď hodnoty prediktora od najmenšieho po najväčšie a pre každé dve susedné hodnoty spočítaj ich priemer. Tieto priemery potom použi ako deliacu hodnotu a_i prediktora X_i a spočítaj kriteriálne štatistiky pre každú hodnotu. Ak je deliaca hodnota a_i prediktora X_i väčšia, alebo rovná hodnote x_i, pozorovanie y_i patrí do pravého uzla, inak do ľavého. Hodnota a_i, pre ktorú je kriteriálna štatistika minimálna je vybraná ako najlepšie možné delenie závislej premennej Y na základe prediktora X_i. Tento postup opakuj pre každý prediktor X_i. Pre každý prediktor X_i tak získame optimálnu deliacu hodnotu a_i. Následne je vybratý prediktor s minimálnou hodnotou kriteriálnej štatistiky a príslušná deliaca hodnota je použitá na rozdelenie pozorovaní do dvoch dcérskych uzlov.
   - V prípade kategoriálneho prediktora X_i sa za účelom najlepšieho rozdelenia prejdú všetky možné kombinácie jednotlivých kategórii prediktora. Podobne ako v spojitom prípade sa použije delenie s najnižšou hodnotou kriteriálnej štatistiky.

2. Rozdeľ súbor na dva dcérske uzly podľa prediktora a hodnoty vybraných v kroku 1.

Opakuj kroky 1 a 2, pokiaľ nie je dosiahnuté jedno z pravidiel zastavenia rastu stromu. Pretože v každom kroku vyberáme z celej množiny prediktorov, môže byť rovnaký prediktor použitý v strome viac krát.

2.4 ZASTAVENIE RASTU STROMU

Maximálna veľkosť stromu je daná veľkosťou a štruktúrou dát. Existujú nasledujúce pravidlá kedy sa rast stromu zastaví.

Rast stromu sa zastaví sám, ak je splnená jedna z nasledujúcich podmienok

1. Uzol obsahuje len jedno pozorovanie
2. Všetky pozorovania v uzle majú rovnakú hodnotu všetkých prediktorov X_i
3. Všetky pozorovanie v uzle majú rovnakú hodnotu nezávislej premennej

V prípade, že chceme rast stromu zastaviť ešte pred splnením podmienok uvedených vyššie, musíme nadefinovať jednu, alebo viac z nasledujúcich hodnôt ako stop kritérium

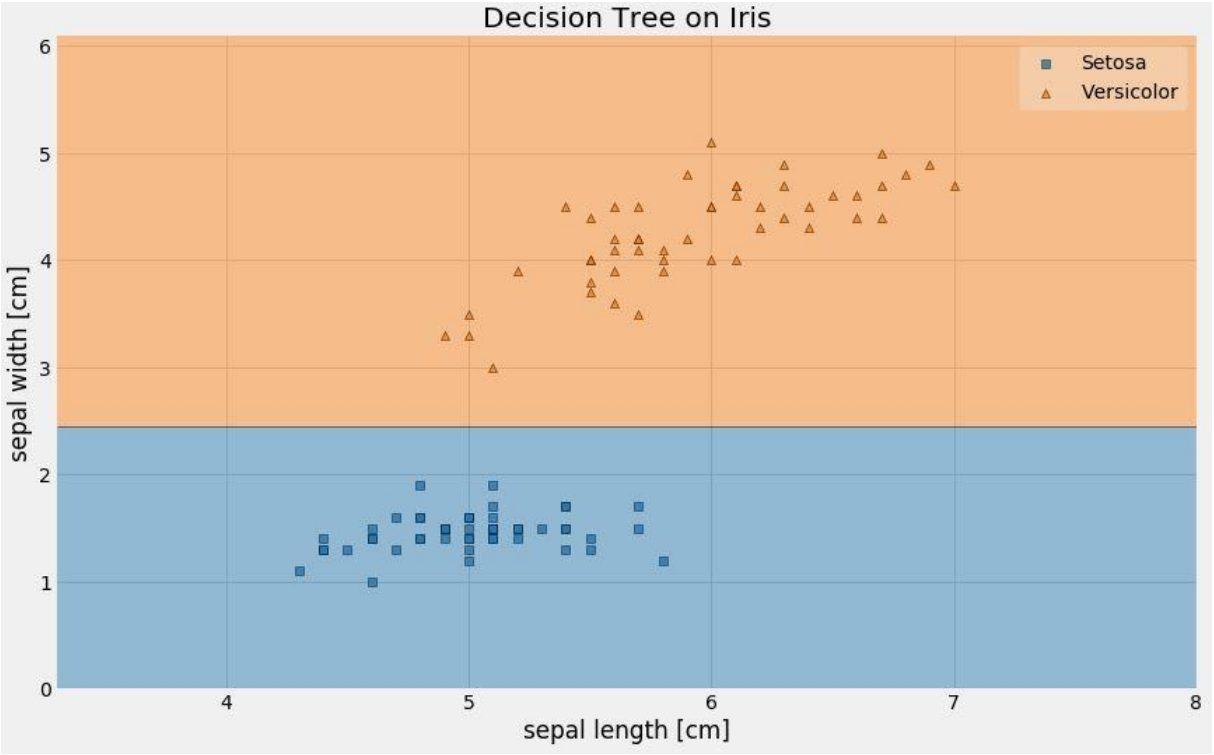
1. Maximálnu hĺbka stromu

2. Minimálny počet pozorovaní v uzle
3. Minimálny počet pozorovaní v liste
4. Maximálny počet listov v strome
5. Minimálna hodnota kriteriálnej štatistiky

Vhodne zvoleným stop kritériom môžeme predísť pretrénovaniu stromu a získať model, ktorý viac generalizuje.

2.5 LINEÁRNE SEPAROVATEĽNÉ DÁTA

V tejto podkapitole opäť použijem rovnaký príklad s lineárne separovateľnými dátami. Výsledok rovnakej klasifikačnej úlohy s použitím rozhodovacieho stromu je možné vidieť na obrázku číslo 9.

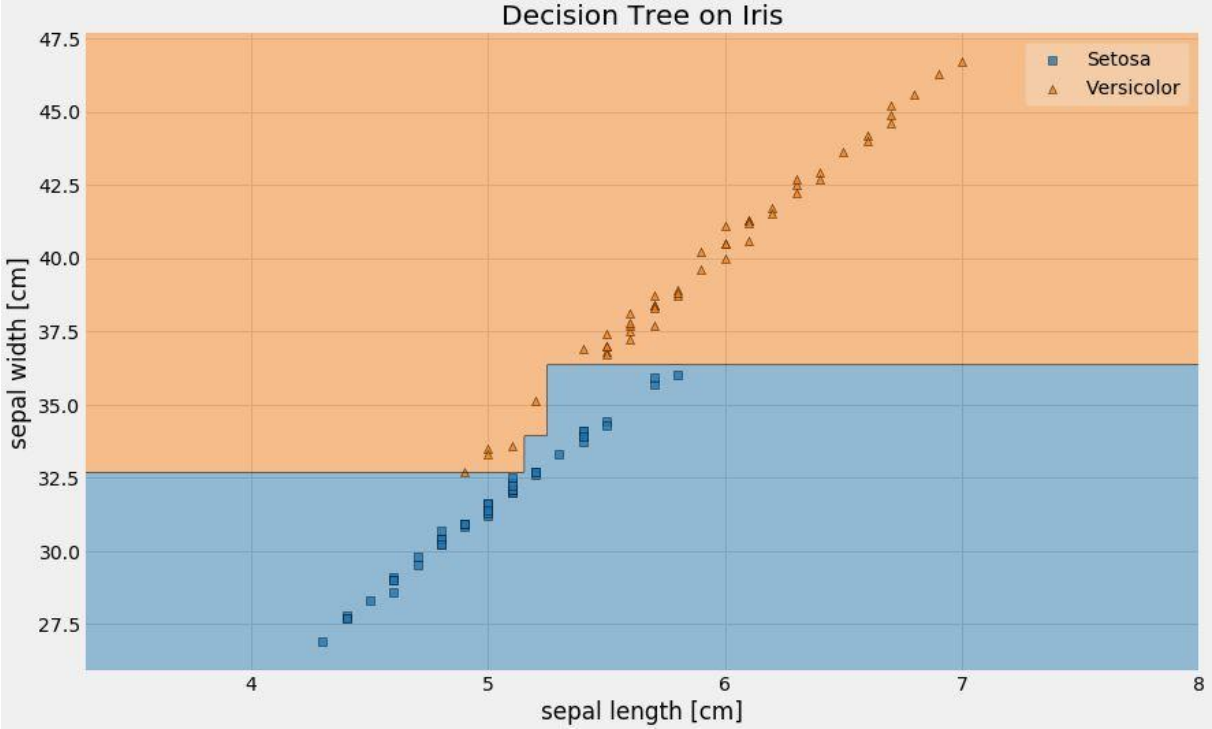


Obrázok 9: Rozhodovací strom – lineárne separovateľné dáta

Z obrázka vidieť, že rozhodovací strom rozdelil priestor prediktorov na dve časti, pričom všetky predikcie sú kategorizované správne. Zaujímavý rozdiel oproti rozdeleniu priestoru logistickou regresiou je ten, že rozhodovací strom delí oblasť vždy kolmo na súradnicový systém. Táto vlastnosť môže extrémne skomplikovať štruktúru stromu pri niektorých (aj lineárne separovateľných) dátach. Takéto dáta je možné vytvoriť napríklad transformáciou

$$sepal\_width := sepal\_width + sepal\_length * 6.$$

Transformované body a klasifikačné oblasti stromu sú na obrázku číslo 10.

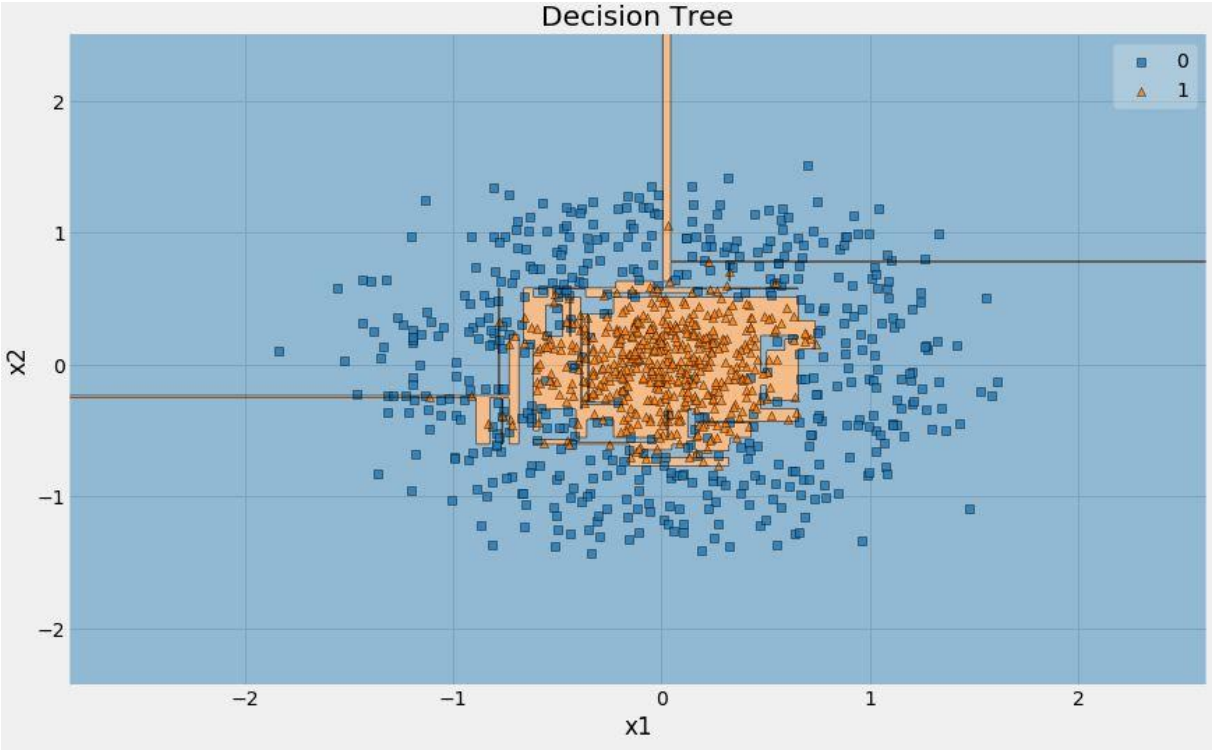


Obrázok 10: Rozhodovací strom – transformované lineárne separovateľné dáta

Z obrázka vidieť, že štruktúra stromu je zložitejšia ako v predchádzajúcom prípade a zároveň klasifikačné oblasti nerozdeľujú dáta ako by sme intuitívne čakali. V takýchto prípadoch je vhodnejšie na modelovanie použiť logistickú regresiu, ktorá túto nevýhodu nemá a s podobne rozloženými dátami si ľahko poradí.

2.6 LINEÁRNE NESEPAROVATEĽNÉ DÁTA

V prípade lineárne neseparovateľných dát použijem rovnaký príklad ako v kapitole o logistickej regresií. Použitie rozhodovacieho stromu na týchto dátach je zobrazené na obrázku číslo 11.



Obrázok 11: Rozhodovací strom – lineárne neseparovateľné dáta

Je vidieť, že rozhodovací strom dokázal klasifikovať dáta oveľa lepšie ako logistická regresia, avšak tiež nie úplne správne. Problémom rozhodovacieho stromu je, že sa snaží klasifikovať aj anomálie v dátach a tým vytvára na pohľad nezmyselné rozhodovacie oblasti. Konkrétny príklad takejto oblasti je vertikálny pruh okolo bodu $(0,1)$. Body spadajúce do tohto pruhu budú klasifikované do kategórie nula, pričom správne patria do kategórie jedna. Tomuto správaniu je však možné predísť vhodne zvolenou podmienkou na zastavenie rastu stromu.

2.7 VÝHODY A NEVÝHODY CART

V tejto podkapitole zhrniem výhody a nevýhody klasifikačných stromov typu CART.

Výhody:

- Nekladú nároky na rozdelenie dát, teda nie je nutné používať transformáciu premenných
- Fungujú so všetkými typmi premenných (kategorické, spojité, ordinálne, atď.)
- Jednoduchá interpretácia výsledkov, graf v tvare stromu
- Algoritmus rastu stromu je odolný voči odľahlým hodnotám
- Je možné použiť korelované prediktory
- Jednoduché porovnanie presnosti s inými algoritmami alebo stromami
- Veľmi rýchla metóda pri klasifikácií nových pozorovaní

Nevýhody:

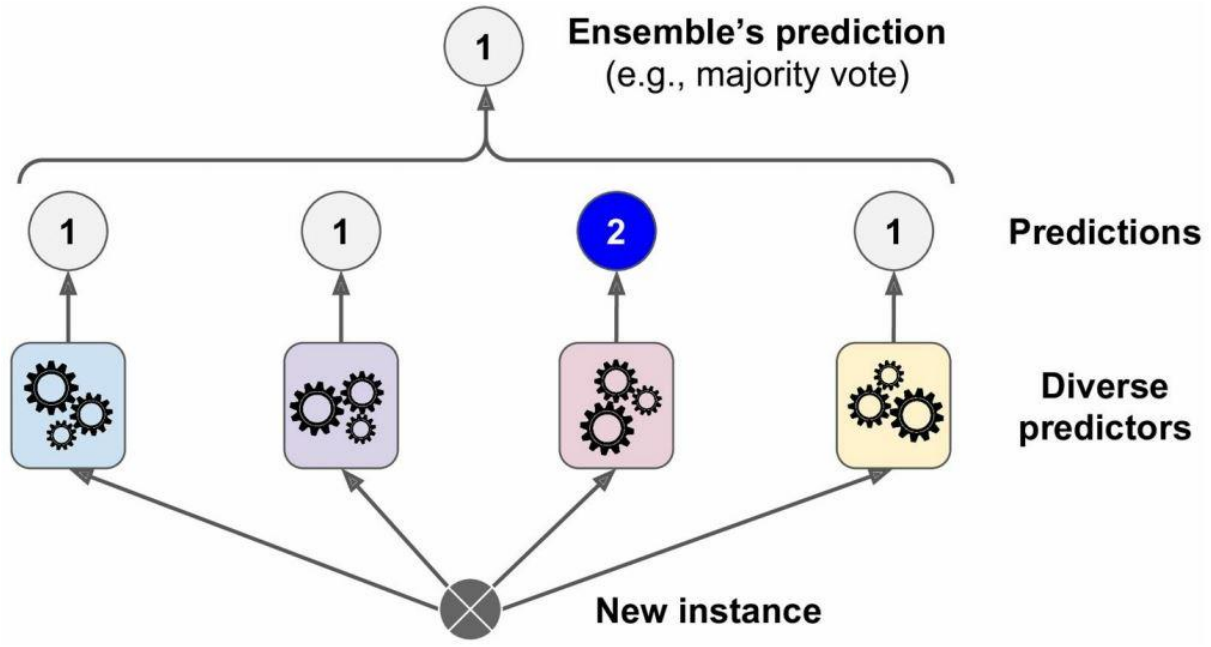
- Nízka stabilita stromu, algoritmus veľmi závisí na trénovacích dátach. Malá zmena trénovacích dát môže viesť k vytvoreniu kompletne odlišného stromu s inými rozhodovacími pravidlami.
- Je nutná opatrnosť pri interpretácií stromu z dôvodu nestability
- Stromy sú nevhodné pre malý počet pozorovaní a zároveň veľký počet prediktorov
- Vytvorenie výsledného stromu závisí na nastavení počiatočných obmedzení rastu stromu, ako maximálna hĺbka a podobne. Tieto parametre majú pre každú úlohu rozdielne hodnoty a tým pádom výsledný strom veľmi závisí na skúsenostiach s nastavovaním parametrov

3 SKUPINOVÉ MODELY (ENSEMBLE)

Pre vysvetlenie skupinových modelov použijem jednoduchý príklad. Predstavme si, že potrebujeme získať odpoveď na jednoduchú otázku. Najjednoduchšia možnosť je opýtať sa túto otázku jedného človeka a považovať jeho odpoveď za správnu a nám postačujúcu. Druhou možnosťou je nespoliehať sa na odpoveď jediného človeka, ale položiť tú istú otázku väčšej skupine ľudí. Je zrejmé, že odpovede ľudí nebudú úplne totožné. Spriemerovaním týchto odpovedí získame vo väčšine prípadov presnejšiu odpoveď na našu otázku. Táto vlastnosť sa nazýva múdrosť skupiny (anglicky wisdom of the crowd). Obdobným spôsobom fungujú skupinové modely, kde ľudia sú nahradení jednotlivými modelmi. Táto technika sa nazýva skupinové učenie (anglicky ensemble learning). Ako príklad môžeme použiť skupinu rozhodovacích stromov natrénovaných na rôznych podmnožinách trénovacieho datasetu. Pretože samostatné stromy sú nestabilný algoritmus (anglicky weak learner), ich predikcie nie sú presné, avšak použitím viacerých stromov dokážeme získať predikcie výrazne presnejšie. Takáto kombinácia rozhodovacích stromov sa nazýva náhodný les (Random forest). Kombináciou nie veľmi presných algoritmov (stromov) sme dostali jeden z najpoužívanejších a najspoľahlivejších modelov vôbec. Náhodné stromy budú detailnejšie vysvetlené ďalej v texte. Teóriu použitú v tejto kapitole som čerpal z [9], [10] a [11].

3.1 HLASOVACÍ MODEL (VOTING CLASSIFIER)

Predstavme si, že máme natrénovaných niekoľko modelov, pričom každý má presnosť približne 80%. Jednoduchý spôsob ako môžeme získať ešte lepší klasifikačný model je agregovať predikcie z týchto modelov a pre každé pozorovanie predikovať kategóriu, ktorá mala najväčší počet hlasov. Pod pojmom hlas rozumieme predikovanú triedu jednotlivých modelov. Pre takýto spôsob hlasovania sa používa pojem tvrdé hlasovanie (anglicky hard voting). Lepšiu predstavu môžeme získať z obrázka číslo 12.



Obrázok 12: Skupinový model – tvrdé hlasovanie (prevzaté z [5])

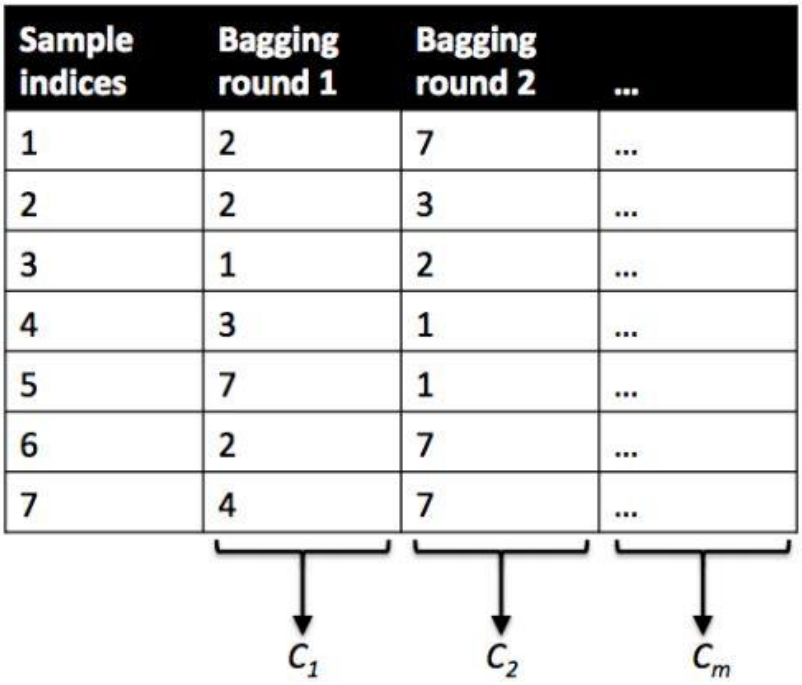
Na obrázku vidieť štyri rôzne natrénované modely. Každý z modelov na vstupe dostane rovnaké pozorovanie. Každý z modelov predikuje, do ktorej kategórie pozorovanie patrí. V našom prípade tri modely predikovali kategóriu jedna a jeden model predikoval kategóriu dva. Keďže väčšina (3/4) modelov predikovala kategóriu jedna, finálna kategória z hlasovacieho modelu bude jedna. Druhý spôsob hlasovania je takzvané mäkké hlasovanie (anglicky soft voting). Princíp je podobný ako pri tvrdom hlasovaní s tým rozdielom, že namiesto predikcie kategórie z každého modelu zoberieme pravdepodobnosť jednotlivých kategórií. Tieto pravdepodobnosti potom spriemerujeme a kategóriu s najvyššou pravdepodobnosťou vyberieme. Tento typ hlasovania býva obvykle presnejší a častokrát doporučený.

Pri hlasovacom modeli je však dôležité spomenúť, že jednotlivé modely by mali byť nezávislé. V prípade závislých modelov by chyba nastávala pri rovnakých pozorovaniach a presnosť oproti jednému jednoduchému modelu by sa nezlepšila.

3.2 BAGGING (BOOTSTRAP AGGREGATING)

Jednou z možností ako získať nezávislé modely je použiť rôzne typy algoritmov (logistická regresia, rozhodovacie stromy, atď.). Druhá možnosť je použiť rovnaký typ algoritmu (napríklad rozhodovacie stromy) a každý model natrénovať na rozdielnej podmnožine trénovacej množiny. Trénovaciu podmnožinu môžeme vytvoriť z pôvodnej trénovacej množiny nasledovne:

Z trénovacej množiny vyberáme náhodne prvky, až pokiaľ nedosiahneme počet prvkov ako v originálnej trénovacej množine. To znamená, že niektoré prvky z pôvodnej trénovacej množiny sa v našej novej množine môžu vyskytovať viackrát a niektoré sa vôbec nemusia objaviť. Tento spôsob vytvárania podmnožín sa nazýva bagging (anglicky bootstrap aggregating). Prvý kto prišiel s touto metódou bol Leo Brieman v roku 1994 [12]. Vo svojom texte ukázal, že bagging dokáže zlepšiť presnosť nestabilných modelov a taktiež znížiť pretrénovanie. Týmto postupom vytvoríme trénovacie podmnožiny pre každý model. Pre lepšiu predstavu poslúži obrázok číslo 13.



Obrázok 13: Bagging (prevzaté z [11])

Pôvodná trénovacia množina z obrázka má sedem pozorovaní. V prvom bootstrapovom výbere vytvoríme opäť množinu so siedmymi pozorovaniami. Môžeme vidieť, že pozorovanie s indexom 2 je v prvom bootstrapovom výbere až trikrát, avšak pozorovania 5,6,7 sa v ňom nenachádzajú. Podľa [12] sa do bootstrapového výberu vôbec nedostane približne 37% pozorovaní z pôvodnej trénovacej množiny. Prvky, ktoré neobsahujú jednotlivé bootstrapové výbery, použijeme na evaluáciu modelu. Tieto pozorovania (anglicky out of bag) model nikdy nevidel a preto presnosť predikcií na týchto pozorovaniach nám napovie o kvalite modelu. Je dôležité si uvedomiť, že každý bootstrapový výber obsahuje iné out of bag pozorovania a kardinalita týchto množín je rôzna. Celkovú presnosť hlasovacieho modelu môžeme získať spriemerovaním presností použitých modelov.

3.3 NÁHODNÉ LESY (RANDOM FOREST)

Lesy môžeme považovať ako určitú nadstavbu nad rozhodovacími stromami. Taktiež sú lesy príkladom skupinového modelu využívajúceho bagging. Lesy odstraňujú nestabilitu rozhodovacích stromov, ale na druhú stranu svojou zložitosťou strácajú výhodu jednoduchej interpretácie. Algoritmy, ktoré nie je možné jednoducho interpretovať sú označované ako „*black-box*“. Napriek tomu je táto technika však vhodná pre datasety s veľkým množstvom prediktorov a malým počtom pozorovaní.

Náhodné lesy môžeme použiť na rôzne typy problémov:

- Klasifikáciu
- Meranie významnosti prediktorov
- Zhlukovanie
- Detekcia odľahlých hodnôt

V tejto práci sa v praktickej časti budem zaoberať prvými dvoma bodmi.

3.3.1 RAST LESA

Náhodný les sa skladá zo stromov $T_1, \dots, T_N$. Pre náhodne lesy sa používajú stromy typu CART, ktoré boli detailne vysvetlené v predchádzajúcej kapitole. Dátový súbor je taktiež nutné rozdeliť na trénovacií súbor s označením L a testovací súbor so označením R. Algoritmus vytvárania lesa vyzerá nasledovne:

1. Vytvor bootstrapový výber L_i z trénovacieho súboru L
2. Vyber náhodne m prediktorov
3. Vytvor strom T_i na bootstrapovom výbere L_i s použitím m náhodne vybraných prediktorov (vytváranie stromu prebieha rovnako ako bolo popísané v metóde CART)
4. Opakuj kroky jedna až tri N- krát. (to jest vytvor N stromov)
5. Spočítaj celkový výsledok klasifikácie na testovacom súbore R s použitím metódy hard votting alebo soft votting

Pre tento algoritmus je však nutné vopred určiť, koľko prediktorov m vybrať a koľko stromov vytvoriť. Tento výber je väčšinou experimentálny a vyžaduje určité skúsenosti. Obvykle sa používajú metódy ako Grid search, alebo Randomized search pre nájdenie hodnôt týchto parametrov, s ktorými model produkuje najmenšiu chybu. Viac o týchto metódach je možné nájsť v [11]. Vzhľadom k tomu, že náhodný les nie je možné pretrénovať, je počet prediktorov jedna z najdôležitejších hodnôt, ktoré treba zvoliť. Počet stromov nás obmedzuje iba časovo a zvolením vysokej hodnoty nič nepokazíme. Preto sa obvykle táto hodnota volí vyššia (často sa odporúča zvoliť počet stromov ako 20-násobok počtu prediktorov) a následne sa znižuje a pri tom sa skúma pokles presnosti modelu.

3.3.2 VÝZNAMNOSŤ PREDIKTOROV

V mnohých prípadoch sa stáva, že dataset obsahuje veľký počet prediktorov. Použitím všetkých dostupných prediktorov presnosť náhodného lesa neznížime, avšak rapídne zvýšime čas na natrénovanie modelu. Na zváženie teda ostáva, či by nebolo možné niektoré prediktory z modelu odstrániť a tým model zjednodušiť a výrazne zrýchliť. V ideálnom prípade sa podarí odstrániť niekoľko prediktorov a neznížiť presnosť modelu. Na výber prediktorov, ktoré budú odstránené a ktoré budú ponechané je možné využiť významnosť premenných vypočítaných pomocou náhodného lesa. Dve najčastejšie implementované metódy sú:

1. Významnosť založená na permutáciách:
   Po vytvorení i-teho stromu na i-tom bootstrapovom výbere sú out of bag pozorovania klasifikované stromom a je spočítaná presnosť klasifikácie. Následne sú hodnoty m-tého prediktora z out of bag pozorovaní permutované a takto upravené pozorovania sú opäť stromom klasifikované. Nasleduje porovnanie presnosti bez permutácie m-tého prediktora a s permutáciou. Pokles v presnosti predikcie, ktorý nastane pri permutovaní, je spriemerovaný cez všetky stromy a použije sa ako miera významnosti m-tého prediktora. V prípade, že je pokles výrazný, tak premenná je významná. Na druhej

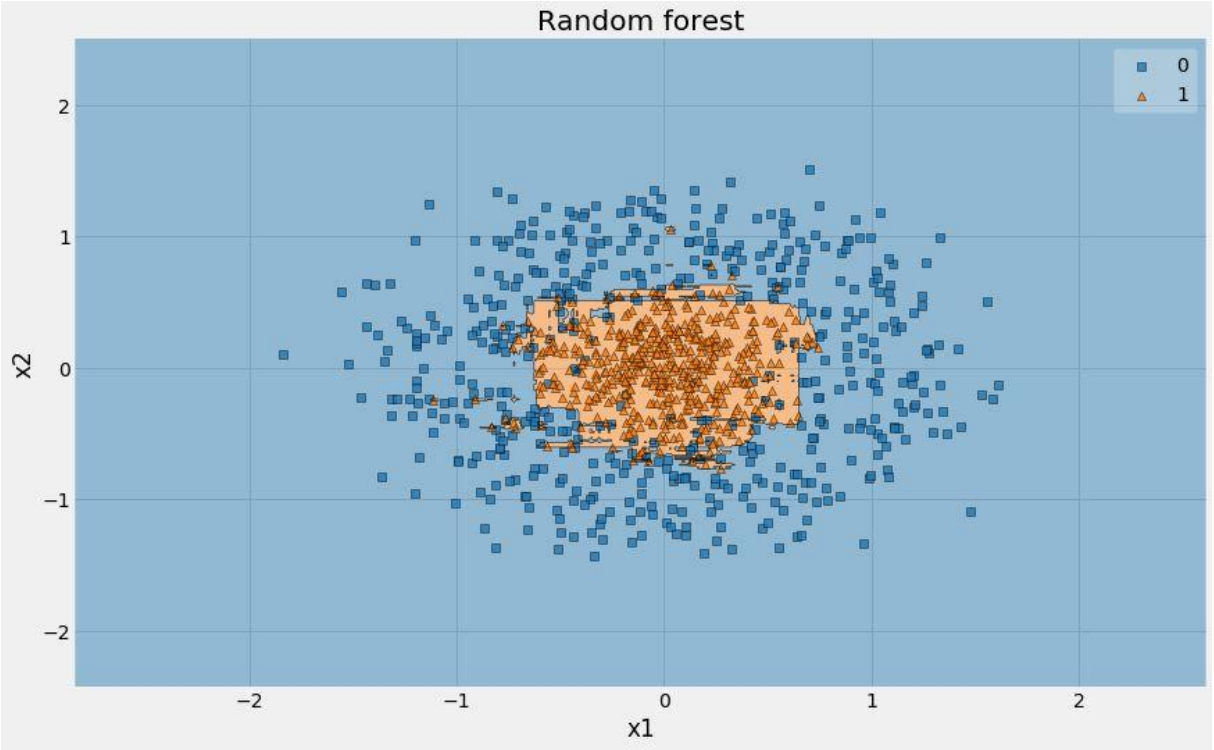
strane, ak permutácia nezmení presnosť klasifikácie, tak je tento prediktor nevýznamný. Čím väčší je rozdiel medzi týmito presnosťami, tým je prediktor významnejší.

2. Významnosť založená na *Gini indexe*:
   V kapitole o rozhodovacích stromoch bol vysvetlený rast stromu s použitím *Gini indexu*. Každé rozdelenie uzla na dva dcérske uzly zníži hodnotu *Gini indexu* vďaka rozdeleniu na základe nejakého prediktora. Súčet poklesov *Gini indexu* pri rozdelení pomocou daného prediktora cez všetky stromy udáva významnosť tohto prediktora. Čím je pokles vyšší, tým je prediktor významnejší.

Jedna z vyššie opísaných metód je následne použitá na výpočet významnosti všetkých prediktorov. Tieto prediktory sú potom zoradené podľa významnosti a hodnoty významnosti transformované do intervalu <0,100>, pričom najvýznamnejší prediktor nadobúda hodnotu 100. Ostáva sa rozhodnúť pod akú hodnotu významnosti prediktory z modelu odstránime. Návod ako túto hodnotu určiť neexistuje, a preto je potrebné vyskúšať rôzne varianty. Cieľom je odstrániť čo najviac prediktorov a iba minimálne znížiť presnosť modelu, v ideálnom prípade ju vôbec neznížiť.

3.4 LINEÁRNE NESEPAROVATEĽNÉ DÁTA

V kapitole o rozhodovacích stromoch bolo možné na lineárne neseparovateľných dátach vidieť, že strom je možné ľahko pretrénovať. Výsledné rozhodovacie oblasti stromu potom nedávali úplne zmysel. V tejto časti použijeme rovnaké dáta a pozrieme sa na výsledok v prípade použitia náhodného lesa. Výsledné rozhodovacie oblasti po natrénovaní náhodného lesa je možné vidieť na obrázku číslo 14.



Obrázok 14: Náhodný les – lineárne neseparovateľné dáta

Z obrázka je vidieť, že model náhodného lesa vytvoril rozhodovacie oblasti podobne ako jediný strom, avšak nie je pretrénovaný. Takýto model je schopný predikovať kvalitnejšie ako rozhodovací strom. Presnosť takéhoto algoritmu teda nezávisí na podmnožine, na ktorej bol daný model natrénovaný a tým sa stáva stabilnejším modelom ako rozhodovací strom. V prípade lineárne separovateľných dát sa model správa podobne ako jediný rozhodovací strom.

3.5 VÝHODY A NEVÝHODY NÁHODNÝCH LESOV

Podobne ako každý model aj náhodné lesy majú svoje výhody a nevýhody.

Výhody:

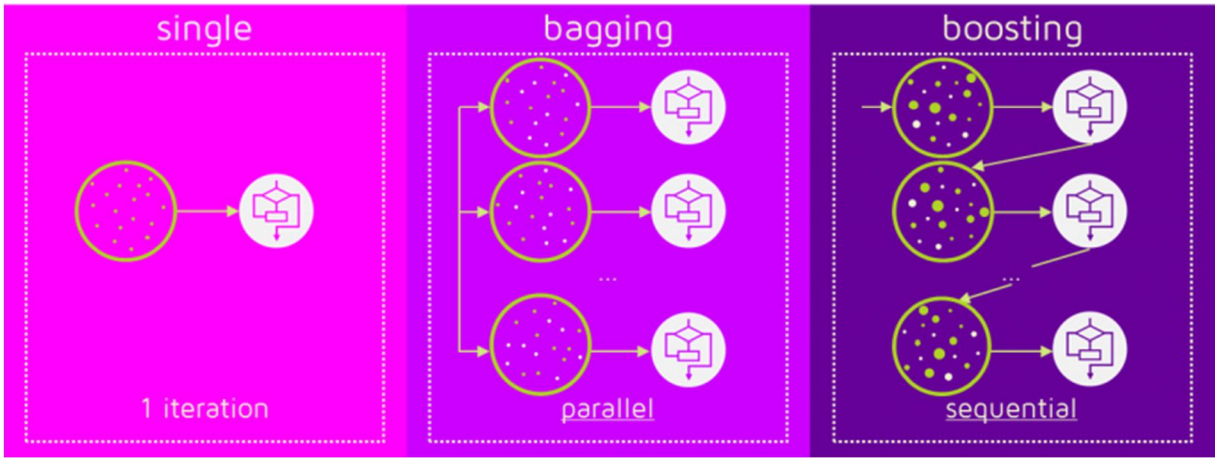
- Jeden z najlepších klasifikačných modelov súčasnosti
- Možnosť použiť na rôzne úlohy (klasifikácia, hľadanie anomálií, zhlukovanie)
- Nie je možné pretrénovať
- Rýchla predikcia
- Implementovaný v mnohých štatistických softwaroch

Nevýhody:

- Nemožná jednoduchá interpretácia („Black-Box“ model)
- Pomalší na natrénovanie

4 BOOSTING

Boosting je sekvenčná metóda zaradená do triedy skupinových modelov. V celom nadchádzajúcom texte budem boosting využívať iba v kombinácií s rozhodovacími stromami, aj keď je možné použiť ktorýkoľvek iný klasifikačný model. Pri použití boostingu je každý model natrénovaný na chybe predchádzajúceho modelu. Na začiatok je dobré vysvetliť princíp boostingu na jednom z najjednoduchších boosting modelov, ktorým je AdaBoost [13]. AdaBoost algoritmus v prvom kroku vytvorí rozhodovací strom s malou hĺbkou. Po klasifikácii pozorovaní a ohodnotení chyby algoritmus zvýši váhu tým pozorovaniam, ktoré boli klasifikované nesprávne a zníži tým, ktoré boli klasifikované správne. Druhý strom je vytvorený na tých istých dátach, avšak s váhami pri jednotlivých pozorovaniach. Ideou tohto kroku je zlepšiť predikciu predchádzajúceho stromu zameraním sa na pozorovania, ktoré predchádzajúci strom klasifikoval nesprávne. Tento proces je možné ďalej opakovať pre ľubovoľný počet stromov. Výsledná predikcia boosting modelu je teda vážená suma predikcií jednotlivých stromov. Pre lepšie pochopenie rozdielu medzi jediným modelom, skupinovým bagging modelom a boostingom poslúži nasledujúci obrázok číslo 15.



Obrázok 15: bagging a boosting porovnanie (prevzaté z [11])

Z obrázka je vidno, že zatiaľ čo bagging model trénuje paralelne viac modelov na rôznych dátach, boosting model trénuje tiež viac modelov, ale na rovnakých dátach s rôznymi váhami, v závislosti od chyby predchádzajúceho modelu.

4.1 XGBOOST (EXTREME GRADIENT BOOSTING)

V tejto kapitole sa budem zaoberať jedným z najlepších modelov, ktoré sa v súčasnosti používajú. Ide o takzvaný XGBoost. Tento model vznikol ako výskumný projekt Tianqi Chena z University of Washington [14]. V súčasnej dobe je model implementovaný v rôznych knižniciach pre programovacie jazyky ako sú napríklad Python, R, Julia, Scala a mnohé iné. Tieto implementácie z neho spravili víťazný model rôznych data mining súťaží po celom svete. Ako príklad predikčnej sily tohto modelu môžem použiť výsledky rôznych data mining súťaží zo stránky Kaggle.com. Z 29 víťazných riešení bol XGBoost použitý v 16-tich prípadoch. Ako druhý najpoužívanejší model sa umiestnili hlboké neurónové siete, použité v 11-tich prípadoch [14]. Prejdime však na matematickú formuláciu modelu.

Majme dataset s n pozorovaniami a m premennými $D = \{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}\}$. XGBoost model využíva súčet K funkcií na predikovanie. Predikciu môžeme zapísať v tvare

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), \qquad f_k \in \mathrm{F}$$

kde

$$\mathrm{F} = \left\{ f(\boldsymbol{x}) = w_{q(\boldsymbol{x})} \mid q\colon \mathbb{R}^m \to \{0,1,\dots,T\}, \boldsymbol{w} \in \mathbb{R}^T \right\}$$

je priestor klasifikačných a regresných stromov (CART). Funkcia q reprezentuje štruktúru stromu, ktorá pozorovaniu priradí index listu, v ktorom pozorovanie skončilo. Počet listov v strome je označených písmenom T. Každej funkcii f_k prislúcha nezávislá stromová štruktúra q s listami o váhach $\boldsymbol{w}$. Budeme používať značenie w_i, ktoré priraďuje skóre listu s indexom i. Keďže hodnota $\hat{y}_i$ leží v intervale $(-\infty, \infty)$, treba ju transformovať do intervalu $(0,1)$. Na túto transformáciu použijeme rovnaký vzťah ako pri modeli logistickej regresie a to

$$\hat{p}_i = \frac{1}{1 + e^{-\hat{y}_i}}$$

kde $\hat{p}_i$ je pravdepodobnosť, že pozorovanie patrí do pozitívnej triedy.

4.2 TRÉNING MODELU

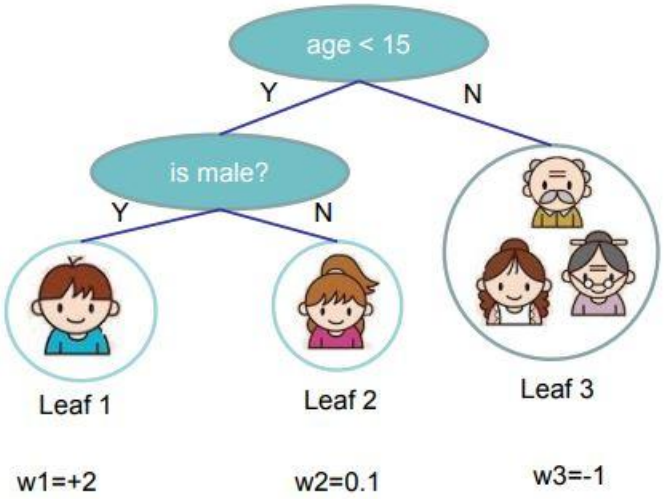
Na rozdiel od logistickej regresie, v ktorej sa snažíme odhadnúť vektor koeficientov **β**, v tomto modeli sa snažíme nájsť funkcie f_k, ktoré minimalizujú účelovú funkciu. Účelová funkcia je obecne v tvare

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

kde prvá suma značí súčet rezíduí cez všetky pozorovania a druhá suma je regularizačný člen, ktorý penalizuje zložitosť jednotlivých stromov v modeli. Funkcia $\Omega(f_k)$ má nasledujúci tvar

$$\Omega(f_k) = \gamma \mathrm{T} + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2$$

avšak toto nie je jediný možný tvar. Pre lepšie pochopenie regularizácie poslúži obrázok číslo 16.



$$\Omega = \gamma 3 + \tfrac{1}{2}\lambda(4 + 0.01 + 1)$$

Obrázok 16: Regularizácia rozhodovacieho stromu (prevzaté z [14])

Na obrázku číslo 16 je jednoduchý strom s troma listami. Počet listov je penalizovaný parametrom γ. Zvýšením tejto hodnoty zabraňujeme vytváraniu nových listov. Parametrom λ penalizujeme váhy jednotlivých listov. Obidva tieto parametre je nutné manuálne nastaviť pred spustením tréningu modelu.

Funkciu $l(y_i, \hat{y}_i)$ je možné zvoliť ľubovoľne podľa povahy riešeného problému. V prípade binárneho klasifikačného modelu sa najčastejšie používa tvar

$$l(y_i, \hat{y}_i) = y_i \ln\left(\frac{1}{1 + e^{-\hat{y}_i}}\right) + (1 - y_i) \ln\left(1 - \frac{1}{1 + e^{-\hat{y}_i}}\right). \qquad (2)$$

Pretože hľadáme vektor funkcií f, ktorý minimalizuje účelovú funkciu a nie vektor hodnôt, nie je možné použiť Stochastic gradient descent. Spôsob akým sa XGB model učí sa nazýva aditívny tréning. Schéma aditívneho tréningu je nasledovná

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(\boldsymbol{x}_i) = \hat{y}_i^{(0)} + f_1(\boldsymbol{x}_i)$$

$$\hat{y}_i^{(2)} = f_1(\boldsymbol{x}_i) + f_2(\boldsymbol{x}_i) = \hat{y}_i^{(1)} + f_2(\boldsymbol{x}_i)$$

...

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(\boldsymbol{x}_i) = \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i).$$

V nultom kroku model predikuje pre každé pozorovanie hodnotu 0. V prvom kroku sa model snaží nájsť funkciu f_1, aby účelová funkcia pre prvý krok

$$L^{(1)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i{}^{(1)}\right) + \sum_{k=1}^{K} \Omega(f_k)$$

nadobúdala minimálnu hodnotu. Všeobecne pre ľubovoľný krok t môžeme účelovú funkciu upraviť na tvar

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i{}^{(t)}\right) + \sum_{k=1}^{K} \Omega(f_k) =$$

$$= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i{}^{(t-1)} + f_t(\boldsymbol{x}_i)\right) + \Omega(f_t) + konštanta.$$

Pretože funkcia

$$l\left(y_i, \hat{y}_i{}^{(t-1)} + f_t(\boldsymbol{x}_i)\right)$$

môže byť ľubovoľná a príliš zložitá, použijeme namiesto nej Taylorov polynóm druhého rádu. Taylorov polynóm druhého rádu funkcie $f(x)$ v bode a má tvar

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2.$$

V našom prípade je $f(x)$ funkcia l, a je predikovaná hodnota v kroku (t-1) a (x − a) je funkcia $f_t(\boldsymbol{x}_i)$, ktorú hľadáme v kroku (t). Funkciu $L^{(t)}$teda môžeme aproximovať nasledovne

$$L^{(t)} \approx \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i{}^{(t-1)}\right) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i) \right] + \Omega(t)$$

kde

$$g_i = \partial_{\hat{y}_i{}^{(t-1)}} l\left(y_i, \hat{y}_i{}^{(t-1)}\right),$$

$$h_i = \partial^2{}_{\hat{y}_i{}^{(t-1)}} l\left(y_i, \hat{y}_i{}^{(t-1)}\right).$$

Odstránením členov nezávislých na f_t dostávame

$$L^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i) \right] + \Omega(t).$$

V tomto kroku si je dobre uvedomiť, že nová účelová funkcia závisí iba na prvej a druhej derivácií pôvodnej účelovej funkcie. Táto vlastnosť môže byť užitočná pri programovaní XGB modelu s vlastnou účelovou funkciou. V prípade zvolenia účelovej funkcie v klasickom najpoužívanejšom tvare (2) sú funkcie g_i, h_i v tvare

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) = \frac{1}{1+e^{-\hat{y}_i}} - y_i$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) = \frac{1}{1+e^{-\hat{y}_i}}\left(1 - \frac{1}{1+e^{-\hat{y}_i}}\right).$$

Definujme množinu pozorovaní, ktorá spadá do listu j ako

$$I_j = \{i \mid q(\boldsymbol{x}_i) = j\}.$$

Účelovú funkciu môžeme vďaka tomuto prepísať na tvar

$$L^{(t)} \approx \sum_{i=1}^{n}\left[g_i f_t(\boldsymbol{x}_i) + \frac{1}{2}h_i f_t^2(\boldsymbol{x}_i)\right] + \Omega(t) =$$

$$= \sum_{i=1}^{n}\left[g_i w_{q(\boldsymbol{x}_i)} + \frac{1}{2}h_i w^2{}_{q(\boldsymbol{x}_i)}\right] + \gamma\mathrm{T} + \lambda\frac{1}{2}\sum_{j=1}^{T} w_j^2 =$$

$$= \sum_{j=1}^{T}\left[\left(\sum\nolimits_{i\in I_j} g_i\right)w_j + \frac{1}{2}\left(\sum\nolimits_{i\in I_j} h_i + \lambda\right)w_j^2\right] + \gamma\mathrm{T}.$$

Vyššie uvedená funkcia je súčtom jednoduchých kvadratických funkcií jednej premennej a dá sa minimalizovať pomocou známych techník. Cieľom je nájsť taký vektor $\boldsymbol{w}$, ktorý minimalizuje účelovú funkciu v kroku t. Pre kvadratickú funkciu platí

$$argmin_x\, Gx + \frac{1}{2}Hx^2 = -\frac{G}{H}, \qquad H > 0$$

$$min_x\, Gx + \frac{1}{2}Hx^2 = -\frac{1}{2}\frac{G^2}{H}.$$

Označme

$$G_j = \left(\sum\nolimits_{i\in I_j} g_i\right)$$

$$H_j = \left(\sum\nolimits_{i\in I_j} h_i\right),$$

potom účelová funkcia prejde na tvar

$$L^{(t)} \approx \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}\left(H_j + \lambda\right)w_j^2\right] + \gamma\mathrm{T}.$$

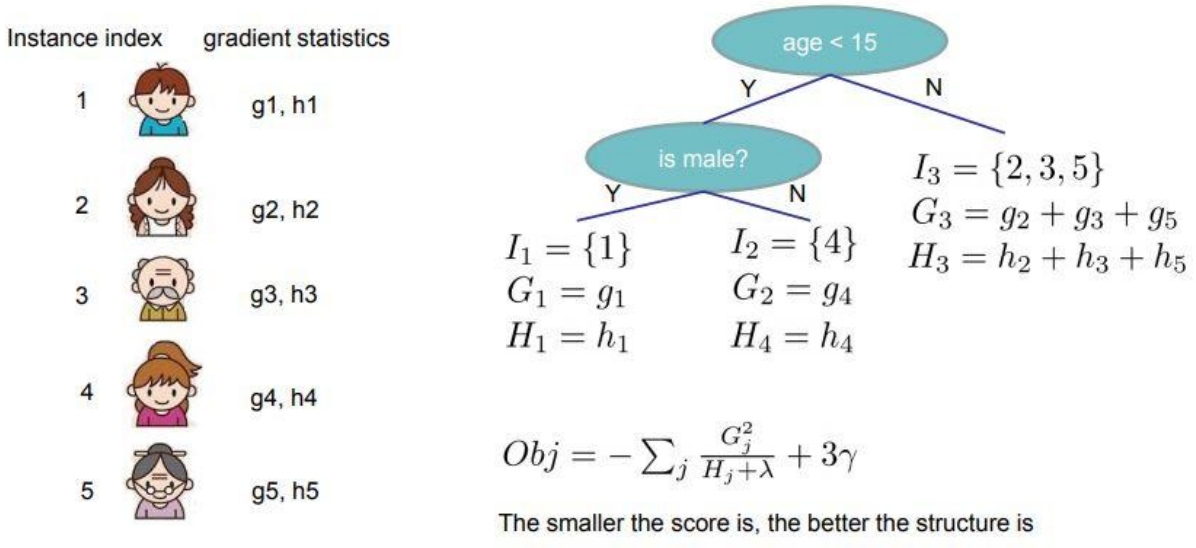
Ďalej predpokladajme, že štruktúra stromu $q(\boldsymbol{x})$ je zafixovaná a už sa nemení. Potom optimálna váha listu j je

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

a účelová funkcia nadobúda hodnotu

$$L^{(t)} \approx -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma \mathrm{T}.$$

Pre lepšiu predstavu poslúži obrázok číslo 17.



Obrázok 17: Výpočet hodnoty účelovej funkcie (prevzaté z [14])

Na obrázku je jednoduchý strom, ktorý vznikol v jednom z K krokov. Na ľavej strane je vidieť päť pozorovaní. Každé z týchto pozorovaní je zaradené stromom do konkrétneho listu s indexom j. Napríklad v liste s indexom $j = 3$, sú pozorovania 2,3 a 5. Pre každý list je nutné spočítať gradientné štatistiky G_j, H_j a následne ich dosadiť do účelovej funkcie. Čím nižšiu hodnotu účelová funkcia nadobúda, tým je štruktúra stromu lepšia.

Pretože nie je možné ohodnotiť všetky možné štruktúry stromu a nájsť štruktúru s najnižšou hodnotou účelovej funkcie, používa sa na vytvorenie stromu rovnaký algoritmus ako sa používal pri náhodných lesoch v kapitole 2. Jediný rozdiel je, že namiesto *Gini indexu* ako kriteriálnej štatistiky sa používa *Gain*. *Gain* môžeme matematicky vyjadriť ako

$$Gain = \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_r^2}{H_r + \lambda} - \frac{(G_l + G_r)^2}{G_l + G_r + \lambda}\right) - \gamma.$$

Gain je možné chápať ako hodnotu, o ktorú sa zlepší (alebo zhorší) účelová funkcia po rozdelení uzla na listy. *Gain* môže nadobúdať kladné hodnoty (v prípade prospešného rozdelenia), ale tiež aj záporné hodnoty. V prípade kategorických premenných nie je nutné meniť štruktúru algoritmu, ale dekódovať kategórie pomocou one-hot encoding [11]. S takto upravenými premennými si už XGB model dokáže poradiť.

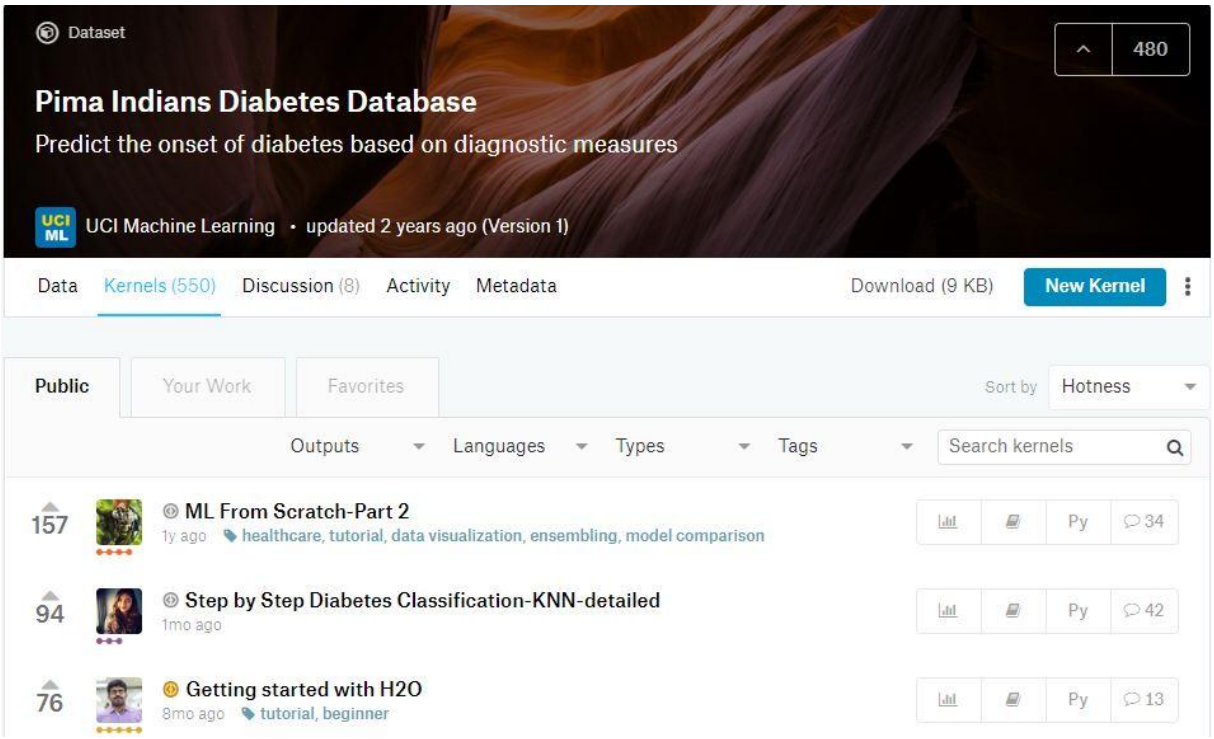
5 PRAKTICKÁ ČASŤ

V tejto časti práce budú aplikované poznatky z teoretickej časti na reálnych dátach. Cieľom je demonštrovať predikčné schopnosti klasifikačných modelov a porovnať ich medzi sebou. Všetky modely boli vyskúšané na voľne dostupnom datasete z portálu Kaggle.com [15].

Kaggle je on-line komunita dátových analytikov a dátových vedcov, vlastnená spoločnosťou Google. Kaggle umožňuje po registrácií a prihlásení voľne pristupovať ku rôznym datasetom na portáli. Tieto datasety je možné analyzovať v ľubovoľne zvolenom programe (R, Python, Julia) a výsledný model zverejniť medzi komunitou. K tomuto modelu sa potom môže vyjadriť každý z členov komunity a tým je možné získať cenné rady a skúsenosti od expertov z celého sveta. Kaggle tiež organizuje verejné súťaže s finančnou výhrou. Spoločnosti, ktoré potrebujú vytvoriť (klasifikačný, regresný, clusteringový...) model, ale nemajú dostatočné vlastné kapacity, zverejnia svoj problém v komunite a vypíšu odmenu za najlepšie riešenie. Každý člen komunity má potom možnosť zapojiť sa do súťaže a poslať svoje riešenie. Najlepšie riešenie je finančne odmenené spoločnosťou a víťazovi taktiež stúpne rank v komunite.

5.1 PIMA INDIANS DIABETES DATABASE

V tejto práci som pracoval s datasetom s názvom „Pima Indians Diabetes Database“ z portálu Kaggle. Dataset je voľne prístupný [15].



Obrázok 18: Kaggle dataset

Tento dataset je v komunite veľmi obľúbený, o čom svedčí aj 550 rôznych analýz od členov komunity. Každú analýzu môžu členovia ohodnotiť a na základe tohto hodnotenia je možné analýzy filtrovať. Z obrázka číslo 18 vidieť, že najlepšie hodnotená analýza je s názvom „ML From Scratch-Part 2“ so 157 pozitívnymi hodnoteniami.

Dataset pochádza z Národného ústavu diabetu, ochorení tráviacej sústavy a obličiek (National Institute of Diabates and Digestive and Kidney Disease). Obsahuje 768 pozorovaní (pacientov), 8 prediktorov a jednu závislú premennú. Pacientov tvoria ženy staršie ako 21 rokov. Prediktory sú rôzne medicínske merania číselného charakteru. Závislá premenná je binárna a indikuje či ma pacient diabetes. V prípade že pacient má diabetes, závislá premenná nadobúda hodnotu 1, inak má hodnotu 0. Cieľom je vytvoriť model, ktorý na základe medicínskych meraní pacienta určí, či pacient má diabetes s čo najväčšou presnosťou.

Použité prediktory, názov prediktora v modeli a vysvetlenie prediktora je v tabuľke nižšie.

Názov prediktora v modeli	Význam
Pregnancies	Počet tehotenstiev
Glucose	Koncentrácia glukózy v plazme dve hodiny po orálnom glukózovom teste
BloodPressure	Diastolický krvný tlak (mm Hg)
SkinThicknes	Hrúbka kožného záhybu - triceps (mm)
Insulin	Množstvo inzulínu v sére (mu U/ml)
BMI	BMI
DiabetesPedigreeF	Výskyt diabetu v rodine
Age	Vek

5.2 POUŽITÝ SOFTWARE

Na analyzovanie dát, vykresľovanie grafov a modelovanie v tejto práci som použil programovací jazyk Python vo verzií 3.6 [16]. Python je moderný programovací jazyk, ktorého popularita stále rastie najmä vďaka jeho jednoduchosti a všestrannej použiteľnosti. Jeho autorom je Guido van Rossum, ktorý ho vymyslel v roku 1989. Jazyk je možné používať na platformách Windows, Linux a Mac. Je open source a taktiež freeware. Na rozdiel od mnohých iných jazykov, ktoré sú kompilačné (C/C++, C#), je Python interpreter. To znamená, že interpreter nevytvára spustiteľný kód (napr. .exe súbor vo Windows), ale na spustenie programu musí byť v počítači nainštalovaný Python.

Hlavné výhody používania Pythonu sú:

- jednoduchá a čitateľná syntax
- nie je nutné dopredu deklarovať typ premenných
- podporuje rôzne programovacie paradigma (objektové programovanie, funkcionálne programovanie, atď.)
- používanie veľkou komunitou programátorov, dobrá dokumentácia

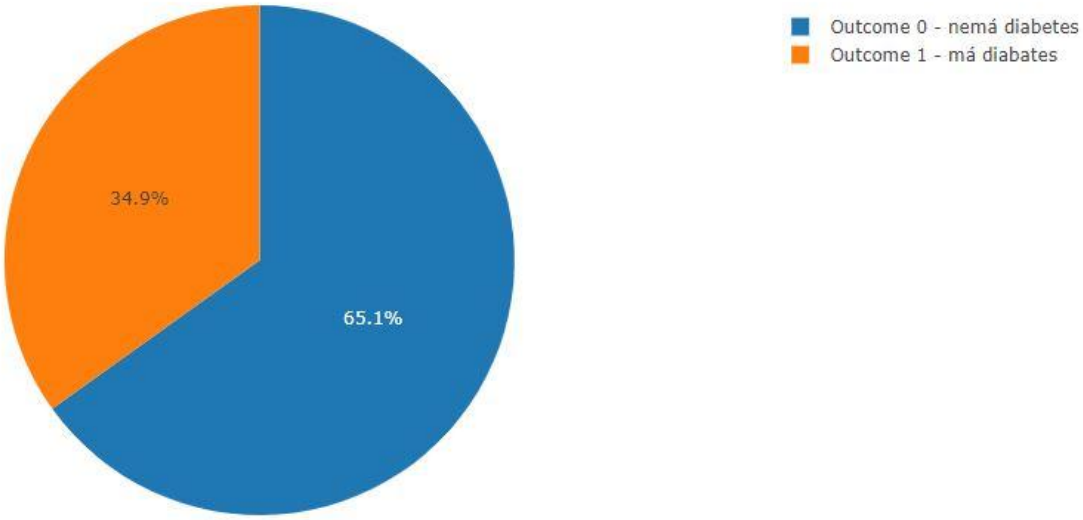
Ako vývojove prostredie som použil Jupyter Notebook [17]. Jupyter Notebook je open-source webová aplikácia, ktorá umožňuje tvorbu a zdieľanie kódu napísaného v Pythone (a 40 iných jazykoch). V tomto prostredí je taktiež možné vytvárať interaktívne dashboardy pre koncového užívateľa vo formáte webovej stránky s príponou html. Používa sa hlavne na prácu s dátami, na rôzne matematické simulácie, štatistické modelovanie a mnohé iné. Samotné knižnice automaticky nainštalované s Pythonom však na prácu s dátami nestačia. Preto som v tejto práci využíval aj nasledujúce externé knižnice:

- Pandas [18] – príprava a manipulácia dát
- Numpy [19] – príprava a manipulácia dát
- Matplotlib [20] – statická vizualizácia dát
- Seaborn [21] – statická vizualizácia dát
- Plotly [22] – interaktívna vizualizácia dát
- Sklearn [23] – modelovanie.

Všetky grafy použité v praktickej časti tejto práce sú interaktívne a nachádzajú sa v priloženej prílohe na CD. K zobrazeniu týchto grafov je však nutné byť pripojený k internetu.

5.3 EXPLORATÍVNA DÁTOVÁ ANALÝZA

V tejto kapitole sa budem venovať exploratívnej analýze dát. Táto analýza slúži na bližšie zoznámenie sa s dátami. V prvom kroku je nutné sa pozrieť koľko pozorovaní v datasete je pozitívnej triedy a koľko negatívnej. Za týmto účelom som vytvoril koláčový graf, ktorý jasne znázorňuje počet pozorovaní pozitívnej triedy a zároveň tento počet vyjadruje aj ako pomer ku všetkým pozorovaniam v percentách. Graf je možné vidieť na obrázku číslo 19.



Obrázok 19: Koláčový graf rozdelenia tried

Z obrázka vidieť, že dataset obsahuje 268 pozorovaní pozitívnej triedy, čo je 34,9% zo všetkých pozorovaní. Z toho vyplýva, že triedy v datasete nie sú rovnovážne rozdelené. Na toto zistenie je nutné si dať pozor pri modelovaní a vyhnúť sa používaniu metódy *predict* v Sklearn knižnici, ale namiesto nej používať *predict_proba.*

Ďalej je nutné zistiť, či dataset obsahuje chýbajúce hodnoty a ako ich prípadne nahradiť. V tomto kroku je možné vidieť aj dátové typy jednotlivých prediktorov a podľa toho sa rozhodnúť, ako tieto prediktory transformovať na čo najvhodnejší tvar (napr. transformovať kategórie na čísla). Dátové typy, počet chýbajúcich hodnôt a využitie pamäte je možne vidieť na obrázku číslo 20.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Obrázok 20: Dátové typy

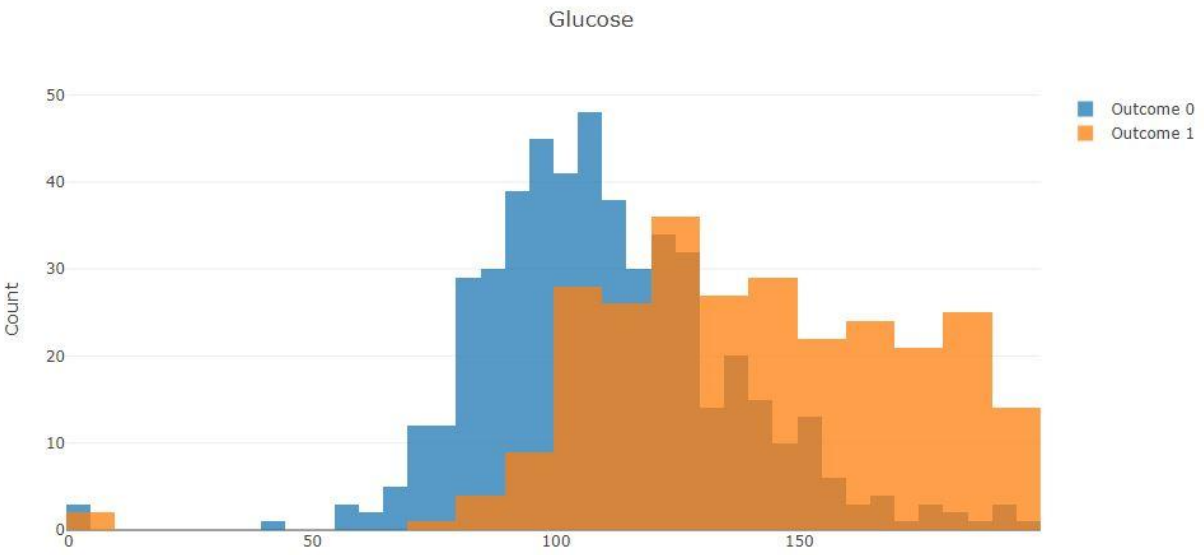
Z obrázka vidieť, že dataset obsahuje 7 celočíselných prediktorov a 2 neceločíselné. Taktiež dataset neobsahuje žiadne chýbajúce hodnoty.

V nasledujúcom kroku sa pozriem na číselné charakteristiky prediktorov ako sú stredná hodnota, smerodajná odchýlka, minimum, maximum, prvý, druhý a tretí kvartil. Z týchto charakteristík je možné získať predstavu o rozdelení dát a o odľahlých hodnotách. Tieto charakteristiky sú na obrázku číslo 21.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
mean	3.85	120.89	69.11	20.54	79.80	31.99	0.47	33.24	0.35
std	3.37	31.97	19.36	15.95	115.24	7.88	0.33	11.76	0.48
min	0.00	0.00	0.00	0.00	0.00	0.00	0.08	21.00	0.00
25%	1.00	99.00	62.00	0.00	0.00	27.30	0.24	24.00	0.00
50%	3.00	117.00	72.00	23.00	30.50	32.00	0.37	29.00	0.00
75%	6.00	140.25	80.00	32.00	127.25	36.60	0.63	41.00	1.00
max	17.00	199.00	122.00	99.00	846.00	67.10	2.42	81.00	1.00

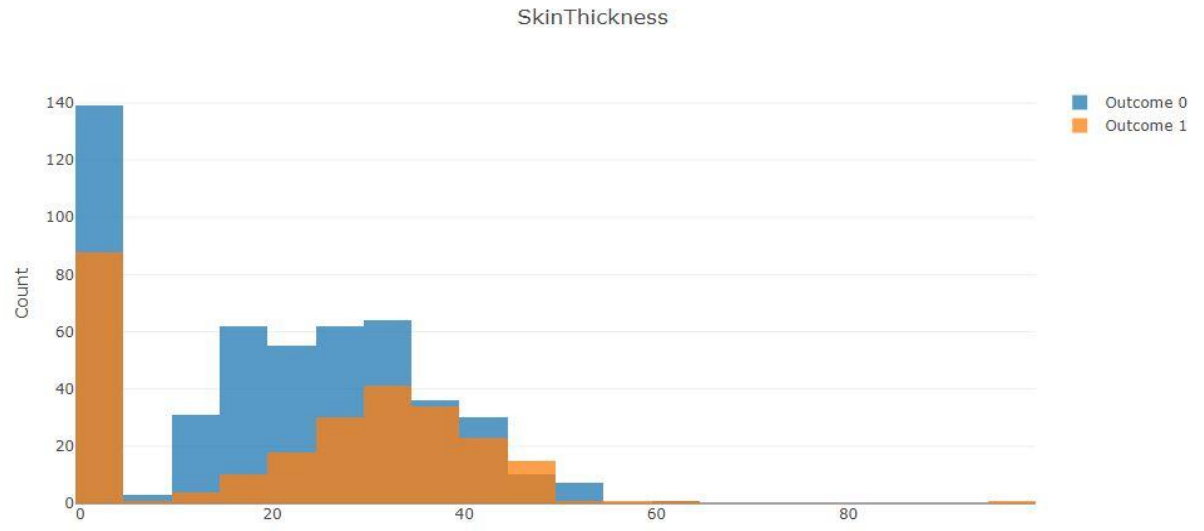
Obrázok 21: Číselné charakteristiky datasetu

Zatiaľ čo číselné charakteristiky pre prediktor Age vyzerajú v poriadku, prediktor Pregnancies už vyzerá podozrivo. Konkrétne, maximálny počet tehotenstiev s hodnotou 17 môže značiť buď odľahlú hodnotu, alebo chybne zadaný údaj. Ako vyhodnotiť takéto pozorovania je však v rukách človeka, čo analyzuje dáta. V ideálnom prípade, ak je to možné, kontaktovať človeka (doktora), ktorý dané dáta zadával a overiť ich platnosť. Vhodnejší spôsob ako zistiť rozdelenia jednotlivých prediktorov je vykresliť histogram pre každý prediktor zvlášť. Vykreslením histogramu pre jednotlivé prediktory v závislosti na závislej premennej Outcome môžeme získať odhad, ktorý prediktor môže byť pre klasifikačný model významný a ktorý naopak nevýznamný. Tento odhad je možné demonštrovať na prediktoroch Glucose a SkinThickness.



Obrázok 22: Prediktor Glucose

Na obrázku číslo 22 je histogram prediktora Glucose v závislosti na závislej premennej Outcome. Je možné si všimnúť, že pacienti s hodnotou prediktora Glucose väčšou ako 120 výrazne viac trpia diabetom. Takéto rozdelenie môže napovedať, že prediktor Glucose bude v modeli významný a neoplatí sa ho z modelu na začiatok vynechávať. Na druhej strane, z histogramu prediktora SkinThickness na obrázku číslo 23 vidieť, že obidva histogramy sa na celej množine prekrývajú a nie je možné nájsť hodnotu prediktora SkinThickness, ktorá by oddeľovala pacientov, ktorý trpia diabetom. Z toho môžeme usúdiť, že prediktor SkinThickness pravdepodobne nebude pre model významný. Pri analyzovaní takýchto histogramov je však nutné mať na pamäti kardinalitu jednotlivých tried. Ako príklad môžeme použiť prediktor SkinThickness s hodnotami od 0 do 4. Môže sa zdať, že pacienti s týmito hodnotami patria skôr do skupiny ľudí bez diabetu, pretože v datasete je viac ako polovica (139 z 227, t.j. 61,2%) ľudí s týmito hodnotami označených ako zdravých. Avšak, keby dataset obsahoval rovnaký počet zdravých ľudí a diabetikov, situácia by mohla byť úplne iná.



Obrázok 23: Prediktor SkinThickness

Iný pohľad na dáta môže byť pomocou krabicových grafov, z ktorých je možné identifikovať odľahlé hodnoty oveľa ľahšie ako z histogramov. Krabicové grafy sa používajú v popisnej štatistike na vizualizáciu dát pomocou ich kvartilov. Krabicová časť diagramu je zdola ohraničená 1. kvartilom (Q1) a zhora 3. kvartilom (Q3). Medzi týmito dvoma hranicami sa nachádza hranica zobrazujúca medián. Fúzy diagramu označujú dolnú a hornú hranicu hodnôt pozorovaní podozrivých z odľahlosti. Tieto hranice môžu byť rôzne v závislosti od použitého softwaru. V našom prípade je dolná a horná hranica stanovená ako

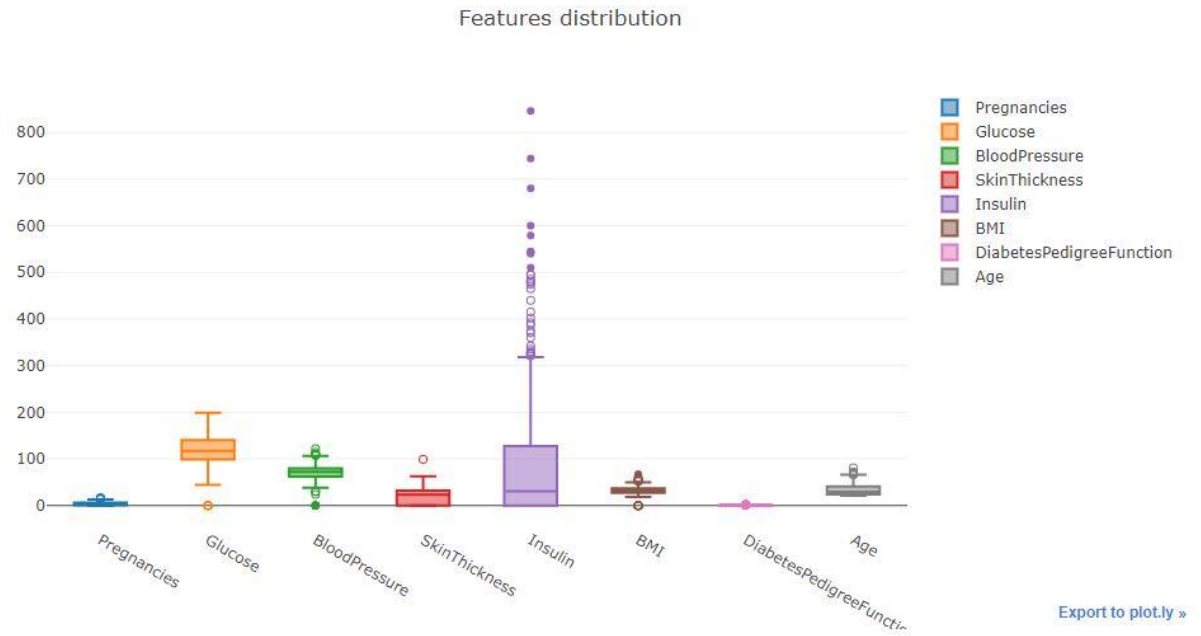
$$Dolná\ hranica = Q1 - 1{,}5 * IQR$$

$$Horná\ hranica = Q3 + 1{,}5 * IQR$$

kde

$$IQR = Q3 - Q1$$

Pozorovania nad hornou hranicu, alebo pod dolnou hranicou sú označené ako podozrivé z odľahlosti a v grafe sú vyznačené nevyplnenou kružnicou. Pozorovania, ktoré sú vzdialené viac ako $3 * IQR$ od prvého kvartilu smerom nadol, alebo od tretieho kvartilu smerom nahor, sú označené ako odľahlé. Tieto pozorovanie sú v grafe znázornené vyplnenou kružnicou. Krabicové grafy všetkých prediktorov sú na obrázku číslo 24.



Obrázok 24: Krabicové grafy

Z grafu vidieť, že najviac pozorovaní podozrivých z odľahlosti a odľahlých pozorovaní obsahuje prediktor Insulin. Naopak, prediktor Glucose obsahuje len jedno pozorovanie podozrivé z odľahlosti s hodnotou 0. Keďže pacient nemôže mať koncentráciu glukózy v plazme nulovú, môžeme toto pozorovanie označiť ako anomáliu. Pozorovanie s touto hodnotou môžeme následne buď odstrániť, nahradiť mediánom, alebo inou štatistikou. Na tieto úpravy však treba myslieť pri zostavovaní modelu a radšej ich vplyv prekonzultovať s človekom (doktorom), ktorý dáta vytvoril.

5.4 MODELOVANIE

V tejto podkapitole sa budem zaoberať modelovaním. Ako prvý krok pri vytváraní štatistického modelu je nutné dataset rozdeliť na dve množiny. Trénovaciu množinu, na ktorej bude model natrénovaný a testovaciu množinu, na ktorej posúdime kvalitu modelu na pozorovaniach, ktoré model nikdy nevidel. V prípade, že by sme dataset nerozdelili, mohlo by sa stať, že sa model naučí konkrétne pozorovania a vykazuje na týchto pozorovaniach perfektnú presnosť. Táto presnosť by však nemusela vôbec zodpovedať presnosti na iných dátach. Testovacia množina sa počas celého procesu vytvárania a ladenia modelu nepoužíva a využije sa až vo chvíli, keď je model hotový a jeho parametre sú odladené. Predikcie modelu na testovacích dátach nám pomôžu získať predstavu o kvalite modelu a jeho schopnosti pri nasadení do produkcie.

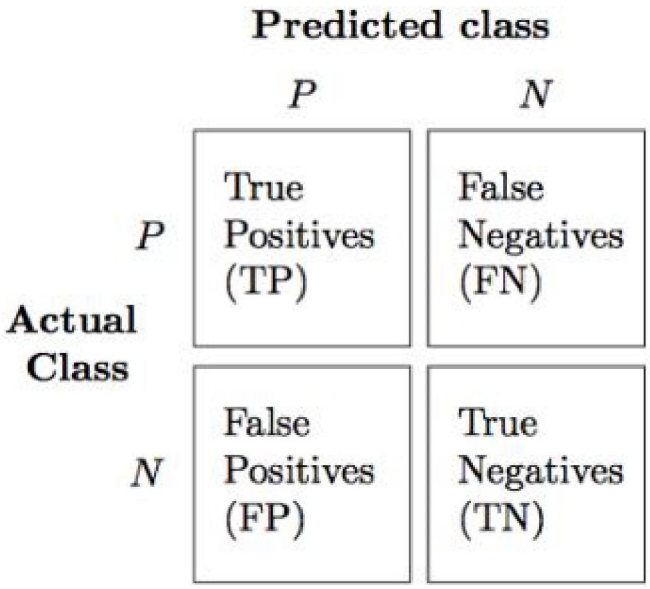
Za týmto účelom som rozdelil dataset na trénovaciu množinu, ktorá obsahuje 75% náhodne vybraných pozorovaní z datasetu a testovaciu množinu, ktorá obsahuje zvyšných 25%. Je však nutné zabezpečiť, aby trénovacia a testovacia množina obsahovali rovnaký pomer pozitívnej triedy ako pôvodný dataset. Na takéto rozdelenie využívam funkciu *train_test_split* zo Sklearn knižnice.

V tejto práci budem testovať a porovnávať nie len modely, ktoré boli popísané v teoretickej časti, ale aj iné, často používané klasifikačné modely. Týmto modelom sa však nebudem venovať detailne, slúžia len na porovnanie so mnou popísanými modelmi. Všetko testovanie uvedené v texte ďalej bude obsahovať modely z tabuľky nižšie.

Názov modelu	Skratka modelu
Logistická regresia	LR
k-Nearest Neighbors	KNN
Klasifikačné a regresné stromy	CART
Gradient Boosting	GBM
Náhodné lesy	RF
Extra stromy	ET
Extreme Gradient Boosting	XGB

Všetky modely okrem Extreme Gradient Boosting modelu sú z knižnice Sklearn. Tabuľka obsahuje dva Gradient boosting modely, pričom každý pochádza z inej knižnice. Ide o totožný algoritmus, avšak inú implementáciu. Do tejto práce som zaradil obidva kvôli porovnaniu ich rýchlosti. Model k-Nearest Neighbors patrí k najjednoduchším klasifikačným modelom. Viac informácií o tomto modeli je možné nájsť v [23]. Model extra stromy je podobný ako náhodné lesy. Rozdiel je v tom, že zatiaľ čo v algoritme náhodných lesov pri raste stromu sa hľadá hodnota prediktora, ktorá najlepšie rozdelí uzol na dva dcérske uzly, pri extra stromoch sa táto hodnota vyberie náhodne. Táto vlastnosť výrazne zrýchľuje rast stromov, ale môže znížiť presnosť. Viac o tomto algoritme je možné nájsť v [23].

Ostáva určiť, čo rozumieme pod slovom presnosť. Pre zadefinovanie tohto pojmu je nutné zaviesť confusion matrix. Confusion matrix pre binárne klasifikačné modely je štvorcová matica, ktorej prvky sú počty True positives (TP), False postitives (FP), False negatives (FN) a True negatives (TN). Ako takáto matica vyzerá je možné vidieť na obrázku číslo 25.



Obrázok 25: Confusion matrix (prevzaté z [5])

Pre vyplnenie Confusion Matrix musíme mať natrénovaný model a oskórovaný testovací dataset. Keďže skutočnú triedu pozorovania poznáme, môžeme spočítať nasledujúce hodnoty:

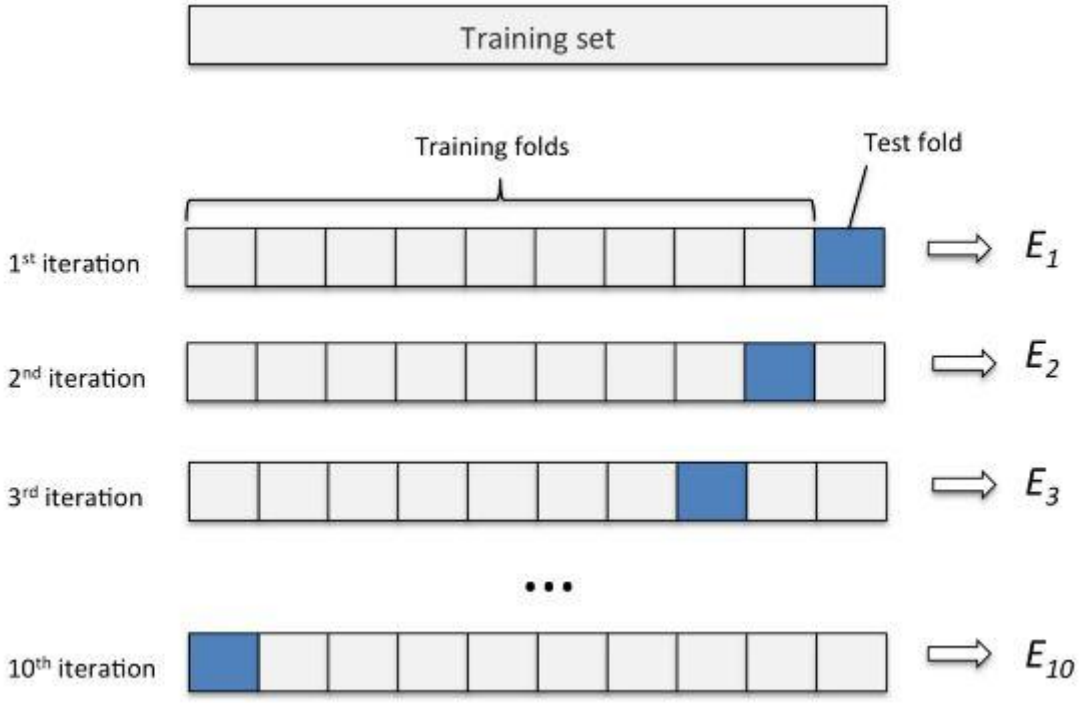
- TP - počet pozorovaní, kde model predikoval pozitívnu triedu a pozorovanie skutočne patrí do pozitívnej triedy
- FN – počet pozorovaní, kde model predikoval negatívnu triedu, ale pozorovanie skutočne patrí do pozitívnej triedy
- FP – počet pozorovaní, kde model predikoval pozitívnu triedu, ale pozorovanie skutočne patrí do negatívnej triedy
- TN – počet pozorovaní, kde model predikoval negatívnu triedu a pozorovanie skutočne patri do negatívnej triedy

Z týchto hodnôt vypočítame celkovú presnosť modelu ako

$$presnosť\ (accuracy) = \frac{TP + TN}{TP + FN + FP + TN}.$$

Hodnoty presnosti sa pohybujú v intervale $\langle 0,1 \rangle$ pričom platí, že čím väčšia hodnota, tým je model kvalitnejší. Nemusí však všeobecne platiť, že model s väčšou presnosťou je lepší z hľadiska využiteľnosti. Z tohto dôvodu sa používajú aj iné spôsoby na overenie kvality modelu (napríklad Precision-Recall krivka, ktorá bude vysvetlená ďalej v texte).

Medzi dobré praktiky pri trénovaní a testovaní modelu patrí k-násobná krížová validácia. Ide o štatistickú metódu, pomocou ktorej dokážeme určiť, ako veľmi ovplyvní tréning modelu na rôznych podmnožinách trénovacej množiny jeho presnosť. V tejto práci používam 10-násobnú krížovú validáciu. Princíp fungovania je nasledovný:
Trénovacia množina sa rozdelí na 10 disjunktných častí. V prvom kroku sa označí desiata časť ako testovacia a na zvyšných deviatich sa natrénuje model. Presnosť modelu sa potom otestuje na testovacej časti. V druhom kroku sa označí ako testovacia deviata časť a na ostatných častiach sa natrénuje ďalší model. Opäť sa presnosť otestuje na testovacej (deviatej) časti. Takto sa postupuje 10 krát. Máme teda 10 modelov, ktoré boli natrénované a otestované vždy na iných častiach trénovacej množiny. Pre lepšiu predstavu poslúži obrázok číslo 26.



Obrázok 26: Krížová validácia (prevzaté z [11])

Presnosti modelov sú označené ako E_i kde $i = 1, \dots, 10$. Priemernú presnosť potom spočítame ako

$$\bar{E} = \frac{1}{10}\sum_{i=1}^{10} E_i$$

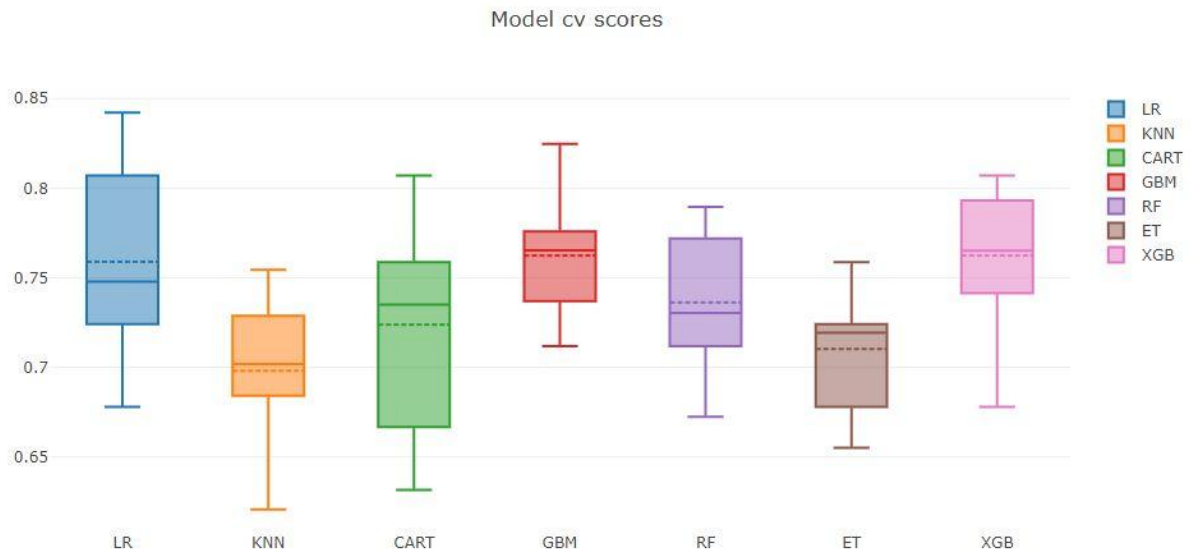
a smerodajnú odchýlku presností ako

$$\sigma = \sqrt{\frac{1}{10}\sum_{i=1}^{10}(E_i - \bar{E})^2}.$$

Porovnaním priemernej presnosti modelov sme schopní povedať, ktorý model má presnejšie predikcie. Porovnaním smerodajných odchýlok zase môžeme posúdiť, ktorý model je stabilnejší vzhľadom na zmenu dát. Priemerné presnosti a smerodajné odchýlky jednotlivých modelov sú v tabuľke nižšie.

Model	$\bar{E}$	σ
LR	0,759	0,050
KNN	0,698	0,042
CART	0,724	0,052
GBM	0,762	0,031
RF	0,736	0,036
ET	0,710	0,032
XGB	0,762	0,036

Z tabuľky vidieť, že najlepšiu priemernú presnosť dosiahli Gradient Boosting modely a tesne za nimi skončila logistická regresia, avšak smerodajná odchýlka presností je výrazne nižšia u Gradient Boosting modelov. Naopak, samostatný rozhodovací strom vykazuje najväčšiu smerodajnú odchýlku a tým aj najväčšiu nestabilitu vzhľadom ku vstupným dátam. Táto vlastnosť rozhodovacích stromov bola však spomenutá v teoretickej časti pri ich nevýhodách. Rozdelenie presností pre jednotlivé modely je možné zobraziť krabicovými grafmi. Tieto grafy sú na obrázku číslo 27.



Obrázok 27: Presnosti modelov

Čiarkovaná vodorovná čiara v krabicových grafoch vyjadruje priemernú presnosť jednotlivých modelov.

Jednou z možností ako zlepšiť predikčné schopnosti modelov je transformácia prediktorov. V prvom kroku som použil štandardizáciu prediktorov a každý z prediktorov som transformoval nasledovne

$$X_{trans} = \frac{X - \bar{X}}{s}$$

kde

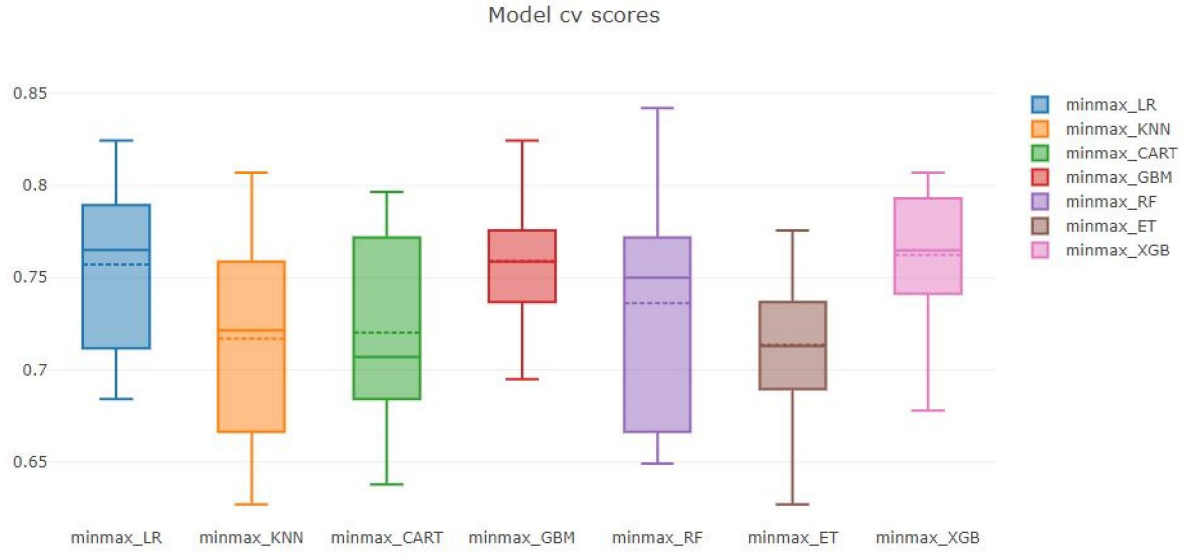
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i,$$

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2}.$$

Po natrénovaní modelov na transformovaných dátach je možne vidieť presnosti predikcií v tabuľke nižšie.

Model	$\overline{E}$	σ
LR	0,762	0,032
KNN	0,709	0,049
CART	0,710	0,031
GBM	0,755	0,034
RF	0,731	0,051
ET	0,717	0,055
XGB	0,762	0,036

Z tabuľky vidieť, že táto transformácia mierne zlepšila presnosť predikcií logistickej regresie a taktiež sa zmenšila smerodajná odchýlka presností. Podobné zlepšenie nastalo aj u KNN modelu. Naopak modely založené na klasifikačných stromoch sa niektoré zlepšili a iné zas zhoršili. Na Extreme Gradient Boosting model transformácia nemala žiadny vplyv. Rozdelenie presností pre jednotlivé modely po tejto transformácii je možne vidieť na obrázku číslo 28.



Obrázok 28: Presnosti modelov

Druhá transformácia prediktorov, ktorú som vyskúšal je transformácia hodnoty každého prediktora do intervalu $\langle 0,1 \rangle$ pričom na hodnotu 1 je transformovaná maximálna hodnota prediktora a na hodnotu 0 minimálna. Túto transformáciu môžeme matematicky popísať ako

$$X_{trans} = \frac{X - \mathrm{X}_{min}}{(X_{max} - X_{min})}$$

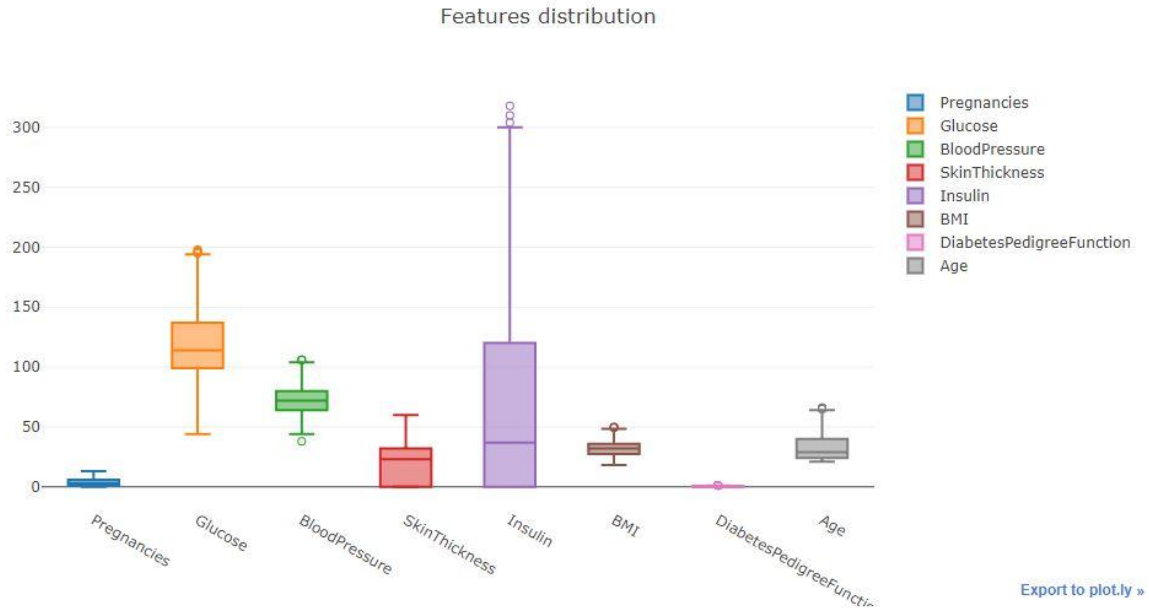
kde X_{min} je minimálna hodnota prediktora a X_{max} je maximálna hodnota prediktora. Presnosti modelov natrénovaných na takto transformovaných dátach sú v tabuľke nižšie.

Model	$\overline{E}$	σ
LR	0,757	0,041
KNN	0,717	0,053
CART	0,720	0,050
GBM	0,759	0,034
RF	0,736	0,058
ET	0,714	0,041
XGB	0,762	0,036

Ako je možné vidieť z tabuľky, niektoré presnosti sa zlepšili oproti predchádzajúcej transformácií a niektoré naopak zhoršili. Preto nie sme schopní rozhodnúť, ktorá transformácia je všeobecne lepšia.

5.4.1 ODSTRÁNENIE ODĽAHLÝCH HODNÔT

Z krabicových grafov jednotlivých prediktorov sme v jednej z predošlých kapitol objavili pozorovania podozrivé z odľahlosti. V tejto časti tieto pozorovania odstránim a natrénujem modely bez týchto pozorovaní. Odstránenie odľahlých hodnôt je však krok, ktorý treba dopredu veľmi dobre zvážiť. V niektorých prípadoch môže byť odstránenie nežiaduce a môže znehodnotiť kvalitu modelu. V našom prípade však odľahlé hodnoty vyzerajú skôr ako chyba pri zadávaní údajov do systému, a preto som sa rozhodol tieto pozorovania odstrániť. Krabicové grafy prediktorov po odstránení odľahlých hodnôt je možné vidieť na obrázku číslo 29.



Obrázok 29: Krabicové grafy po odstránení odľahlých hodnôt

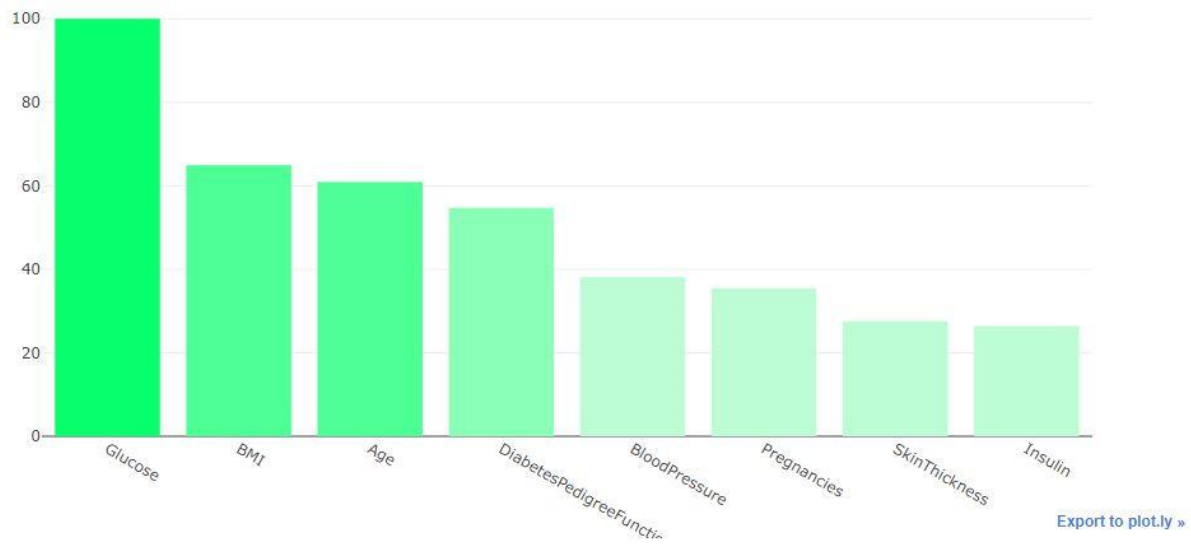
Odstránením odľahlých hodnôt sme prišli o 16,8% pozorovaní a náš dataset sa zmenšil z pôvodných 768 pozorovaní na 639. Na takto upravený dataset následne aplikujem transformácie prediktorov popísané v predchádzajúcom texte a natrénujem modely. Presnosť modelov po odstránení odľahlých hodnôt a aplikovaní transformácií je v tabuľke nižšie.

Model	Štand. $\overline{E}$	Štand. σ	MinMax$\overline{E}$	MinMax. σ
LR	0,785	0,058	0,775	0,052
KNN	0,754	0,034	0,743	0,031
CART	0,718	0,047	0,731	0,060
GBM	0,775	0,066	0,773	0,064
RF	0,739	0,057	0,754	0,066
ET	0,733	0,060	0,750	0,045
XGB	0,777	0,074	0,777	0,074

Z tabuľky vidieť, že odstránenie odľahlých hodnôt zlepšilo presnosť modelov pri oboch transformáciách.

5.4.2 VÝZNAMNOSŤ PREDIKTOROV

V kapitole o náhodných lesoch bolo uvedené, že ich je možné použiť aj ako nástroj na zistenie významnosti prediktorov. Prediktory je následne možné zoradiť podľa významnosti od najväčšej po najmenšiu. Najvýznamnejší prediktor nadobúda hodnotu 100. Takto zoradené prediktory je možné vidieť na obrázku číslo 30.



Obrázok 30: Významnosť prediktorov

Z obrázka je možné vidieť, že najvýznamnejší prediktor je podľa náhodného lesa prediktor Glucose. Tým sa potvrdil aj náš predpoklad z exploratívnej analýzy, že tento prediktor bude významný. Taktiež sa potvrdil náš predpoklad, že prediktor Skin Thickness nebude príliš významný.

5.4.3 ZNÍŽENIE DIMENZIE DATASETU

V predchádzajúcej podkapitole sme zistili významnosť jednotlivých prediktorov. Tieto významnosti môžeme použiť na zníženie dimenzie datasetu. Cieľom je odstrániť niektoré prediktory a tým úlohu zjednodušiť, ale zároveň neznížiť presnosť modelov. Jedna z možností

ako prediktory odstraňovať je využiť pri tom ich významnosť. V prvok kroku sa odstráni z datasetu prediktor s najmenšou hodnotou významnosti. Následne sa natrénuje model na datasete bez tohto prediktora a vypočíta sa jeho presnosť. Ak presnosť neklesla, odstránime prediktor s druhou najnižšou hodnotou. Opäť natrénujeme model, teraz však na datasete bez dvoch najmenej významných prediktorov. Vypočítame presnosť modelu. Ak presnosť klesla, použijeme predchádzajúci model s jedným odstráneným prediktorom. V prípade, že neklesla postupujeme v odstraňovaní ďalej. V našom prípade bolo možné z datasetu odstrániť štyri najmenej významné prediktory, viď. obrázok číslo 31, bez straty presnosti modelov. Po transformáciách datasetu bez týchto štyroch prediktorov a natrénovaní modelov na takto upravenom datasete boli presnosti nasledovné.

Model	Štand. $\overline{E}$	Štand. σ	MinMax$\overline{E}$	MinMax. σ
LR	0,776	0,059	0,783	0,053
KNN	0,780	0,056	0,787	0,068
CART	0,701	0,042	0,737	0,054
GBM	0,760	0,068	0,760	0,067
RF	0,764	0,066	0,755	0,047
ET	0,772	0,078	0,774	0,060
XGB	0,766	0,069	0,766	0,069

Z tabuľky vidieť, že presnosť modelov sa nijako výrazne nezhoršila, pričom zložitosť úlohy sa výrazne znížila. Takéto zníženie dimenzie dát môže mať za následok významné zníženie nákladov na vyšetrenie pacienta znížením počtu vyšetrení, ktoré musí pacient absolvovať v prípade, že mu chce lekár diagnostikovať diabetes. Taktiež je možné vidieť, že pomocou MinMax transformácie sú presnosti takmer pri všetkých modeloch vyššie ako pri transformácií štandardizáciou. Preto v ďalšom texte budem používať dataset iba so štyrmi najvýznamnejšími prediktormi (Glucose, BMI, Age, DiabetesPedigreeFunction) transformovanými pomocou MinMax transformácie.

5.4.4 LADENIE HYPERPARAMETROV MODELOV

Hyperparametre modelu sú parametre, ktoré nie je možné odhadnúť alebo vypočítať zo vstupných dát modelu. Tieto parametre musia byť nastavené ešte pred trénovaním modelu. V predchádzajúcich prípadoch sme v modeloch vždy používali základné nastavenia hyperparametrov a ich hodnotu sme nemenili. Keďže už máme vybraté prediktory a transformácie, ktoré použijeme pri modelovaní, ostáva nám vyladiť práve tieto hyperparametre. Pretože pre rozdielne datasety bude presnosť modelu najvyššia s rozdielnymi nastaveniami hyperparametrov je potrebné pre každý dataset nájsť hyperparametre zvlášť. Neexistuje však žiadny návod ako tieto parametre pri konkrétnych úlohách zvoliť.

Jedným z možných riešení je použiť metódu s názvom Grid Search. Vstupy do metódy sú pevne zvolené hodnoty hyperparametrov modelu. Metóda vyskúša všetky možné kombinácie týchto parametrov a pre každú kombináciu natrénuje model, pričom jeho presnosť spočíta pomocou k-násobnej krížovej validácie. Model s najväčšou priemernou presnosťou z krížovej validácie je vybraný a jeho hyperparametre sú označené ako najlepšie možné z testovaných možností na danom datasete.

Všetkým modelom v tejto práci som ladil hyperparametre pomocou Grid Search metódy. Po vybratí najlepšej kombinácie hyperparametrov pre každý model som jeho presnosť testoval na testovacej podmnožine, aby som overil ako sa správa model na dátach, ktoré neboli nikde predtým použité. Týmto spôsobom môžem získať predstavu, aký presný model som schopný dodať do produkčného riešenia. Množina hyperparametrov pre jednotlivé modely a ich presnosť je nasledovná:

LR

Parameter	Hodnoty
C	0,01;0,02;...;1
penalty	L_1, L_2

Hyperparameter **C** značí silu regularizácie modelu. Čím vyššiu hodnotu nadobúda, tým je model menej regularizovaný. Parameter **penalty** značí typ regularizácie.

Ako optimálne hodnoty hyperparametrov pre logistickú regresiu boli nájdené hodnoty **C = 0,57** a **penalty = L_2**. S týmito parametrami mal model priemernú presnosť z 10-násobnej krížovej validácie 0,787. Presnosť na testovacej podmnožine bola 0,768.

KNN

Parameter	Hodnoty
neighbors	1,2,...,30

Hyperparameter **neighbors** určuje na základe koľkých najbližších pozorovaní je určená trieda pozorovania. V prípade, že v okolí pozorovania ktoré model predikuje je viac ako 50% pozorovaní pozitívnej triedy, model kategorizuje pozorovanie ako pozitívne. Práve hyperparameter **neighbors** určuje, koľko najbližších pozorovaní má model zohľadniť pri predikcii. Optimálna hodnota tohto parametra je v našom prípade **neighbors = 11**. Z krížovej validácie model dosahoval priemernú presnosť 0,793. Presnosť na testovacej množine bola 0,793.

CART

Parameter	Hodnoty
max_depth	3,5,None
max_features	1,2,3,4
min_samples_leaf	1,2,3,4
criterion	Gini, Entropy

Hyperparameter **max_depth** vyjadruje najväčšiu možnú hĺbku stromu, **max_features** vyjadruje počet premenných, podľa ktorých sa strom rozhoduje pre najlepšie delenie uzla, **min_samples_leaf** vyjadruje minimálny počet pozorovaní v liste a **criterion** vyjadruje použitú kriteriálnu štatistiku pri raste stromu. Optimálne hodnoty týchto parametrov sú **max_depth = 5**, **max_features = 4**, **min_samples_leaf = 1** a **criterion = Gini**. Z krížovej

validácie model dosahoval priemernú presnosť 0,768. Presnosť na testovacej množine bola 0,735.

RF

Parameter	Hodnoty
max_depth	3,5,None
n_estimators	200,500
min_samples_split	2,3,4
criterion	Gini, Entropy

Hyperparametre **criterion** a **max_depth** majú rovnaký význam ako pri rozhodovacom strome. Hyperparameter **min_samples_split** určuje minimálny počet pozorovaní v uzle, kedy sa uvažuje delenie. **N_estimators** určuje počet rozhodovacích stromov použitých v náhodnom lese. Optimálne hodnoty týchto parametrov sú **max_depth = 3**, **n_estimators = 200**, **min_samples_split = 2** a **criterion = Gini**. Z krížovej validácie model dosahoval priemernú presnosť 0,78. Presnosť na testovacej množine bola 0,768.

ET

Parameter	Hodnoty
max_depth	3,5,None
n_estimators	200,500
min_samples_split	2,3,4
criterion	Gini, Entropy

Pre hyperparametre extra stromov boli použité rovnaké hodnoty ako pri náhodnom lese. Optimálne hodnoty týchto parametrov sú **max_depth = 5**, **n_estimators = 200**, **min_samples_split = 3** a **criterion = Gini**. Z krížovej validácie model dosahoval priemernú presnosť 0,793. Presnosť na testovacej množine bola 0,781.

Gradient Boosting

Parameter	Hodnoty
learning_rate	0,01;0,05;0,1;0,5;0,75;1
n_estimators	50,100,150,200,250,300

Hyperparameter **learning_rate** vyjadruje rýchlosť učenia modelu, **n_estimators** vyjadruje počet rozhodovacích stromov v modeli. Optimálne hodnoty týchto parametrov sú **learning_rate = 0,05** a **n_estimators = 200**. Z krížovej validácie model dosahoval priemernú presnosť 0,768. Presnosť na testovacej množine bola 0,731.

XGB

Parameter	Hodnoty
learning_rate	0,01;0,05;0,1;0,5;0,75;1
n_estimators	50,100,150,200,250,300

Pre XGB model boli použité rovnaké hyperparametre ako v Gradient Boosting modeli. Optimálne hodnoty týchto parametrov sú **learning_rate = 0,01** a **n_estimators = 200**. Z krížovej validácie model dosahoval priemernú presnosť 0,764. Presnosť na testovacej množine bola 0,75.

Za zmienku však stojí porovnanie rýchlostí Gradient Boosting modelu a XGB modelu. Napriek tomu, že sa jedná o rovnaký algoritmus, tréning modelu aj s krížovou validáciou trval Gradient Boosting modelu 22 sekúnd, avšak XGB model to isté zvládol za 8 sekúnd. Pri takto malom datasete je rozdiel zanedbateľný, ale pri použití na reálnych dátach obsahujúcich niekoľko miliónov pozorovaní bude časový rozdiel priepastný. Z tohto dôvodu je v praxi oveľa viac využívaný XGB model.

5.5 VYHODNOTENIE KVALITY MODELOV

Na vyhodnotenie kvality modelov je možné použiť aj iné kritériá ako presnosť. V tejto kapitole uvediem ďalšie dva rôzne spôsoby ako určiť kvalitu modelu a ako vybrať najvhodnejší model.

5.5.1 ROC KRIVKA, AUC

Skratka ROC je z anglického Receiver Operating Characteristic. Táto metóda sa často používa pri vyhodnocovaní kvality klasifikačných modelov s binárnou závislou premennou. Pre zostrojenie ROC krivky je však nutné definovať pojmy ako

$$senzitivita = \frac{TP}{TP + FN} = \frac{TP}{T}$$

$$špecificita = \frac{TN}{TN + FP}$$

$$precíznosť = \frac{TP}{TP + FP}$$

$$1 - špecificita = \frac{FP}{F}$$

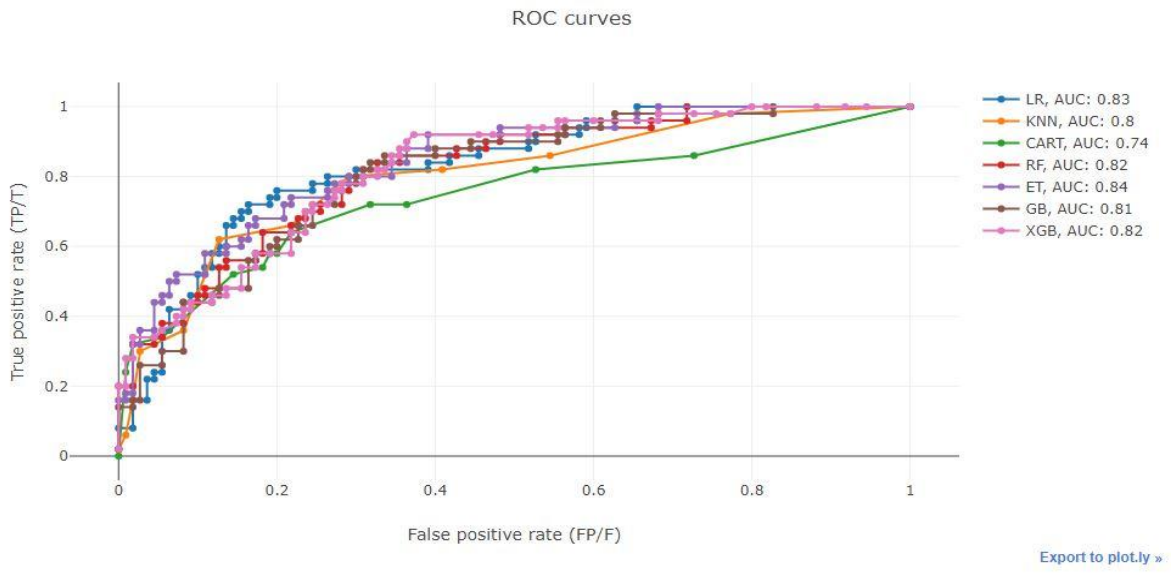
kde hodnoty TP, FN, TN a FP sú vypočítané z Confusion Matrix. V ROC grafe je na x-ovej osi vynesená hodnota 1 – špecificita a na y-ovú os je vynesená senzitivita. Z modelu dostaneme predpoveď výsledkov (pozitívnu alebo negatívnu triedu pre každé skórované pozorovanie). Z týchto výsledkov vypočítame potrebné štatistiky a bod vynesieme do dvojrozmerného ROC grafu. Bod [0,0] predstavuje model, ktorý všetky pozorovania klasifikuje ako negatívne. Bod [1,0] predstavuje ideálny model, ktorý všetky pozorovania pozitívnej triedy klasifikoval ako

pozitívne a všetky pozorovania negatívnej triedy klasifikoval ako negatívne. Naopak bod [0,1] predstavuje model, ktorý všetky pozorovania kategorizoval naopak. Takýto model je však jednoduché upraviť na ideálny model tým, že ako predikciu budeme uvažovať opačnú triedu ako model predikoval. Bod [1,1] predstavuje model, ktorý všetky pozorovania klasifikoval ako pozitívne. Z toho vyplýva, že čím je model bližšie vynesenými hodnotami ku bodu [1,0], tým je kvalitnejší. Body na diagonále (y = x) grafu označujú náhodnú predpoveď, a preto takýto model je nevhodný na predikovanie, pretože nemá žiadnu informáciu o predikovanej veličine. Veľkou výhodou ROC grafu je, že nie je závislý na proporciách pozitívnej a negatívnej triedy. V prípade zmien týchto proporcií v datasete ROC krivka ostane neovplyvnená touto zmenou.

V texte vyššie som písal o jednom bode v grafe, ktorý reprezentuje skúmaný model. Tento bod sme získali z pravdepodobností pozitívnej triedy, ktoré model predikoval. Pozorovanie sme kategorizovali do pozitívnej triedy, ak model predikoval pravdepodobnosť vyššiu ako 0,5. S takto kategorizovanými pozorovaniami sme následne vypočítali štatistiky (senzitivitu, špecificitu) a bod vyniesli do grafu. Keďže graf ROC je krivka a nie jediný bod, je nutné vypočítať aj ostatné body krivky. Tieto body je možné vypočítať pomocou posúvania hranice pravdepodobnosti, kedy model predikuje pozitívnu triedu. Keďže model predikuje pravdepodobnosti s hodnotami z intervalu $\langle 0,1 \rangle$, budeme používať tiež hranice z tohto intervalu. Z intervalu vyberieme ľubovoľný počet hodnôt (podľa toho, z koľkých bodov chceme vytvoriť ROC krivku) a každú z týchto hodnôt použijeme ako hranicu pravdepodobnosti, nad ktorou pozorovanie kategorizujeme do pozitívnej triedy. Pre takto kategorizované pozorovania vypočítame potrebné štatistiky a bod vynesieme do grafu. Tento postup sa opakuje pre každú hraničnú hodnotu. Spojením všetkých vynesených bodov vzniká ROC krivka.

Plocha pod touto krivkou sa označuje ako AUC (anglicky Area Under Curve). AUC môže nadobúdať hodnoty z intervalu $\langle 0,1 \rangle$. Keďže plocha pod diagonálou je rovná 0,5 a vyjadruje náhodné predikcie, nemali by sme používať model s hodnotou AUC < 0,5. Čím je hodnota AUC väčšia, tým je model kvalitnejší.

ROC krivky a AUC skóre pre testované modely je možné vidieť na obrázku číslo 31.



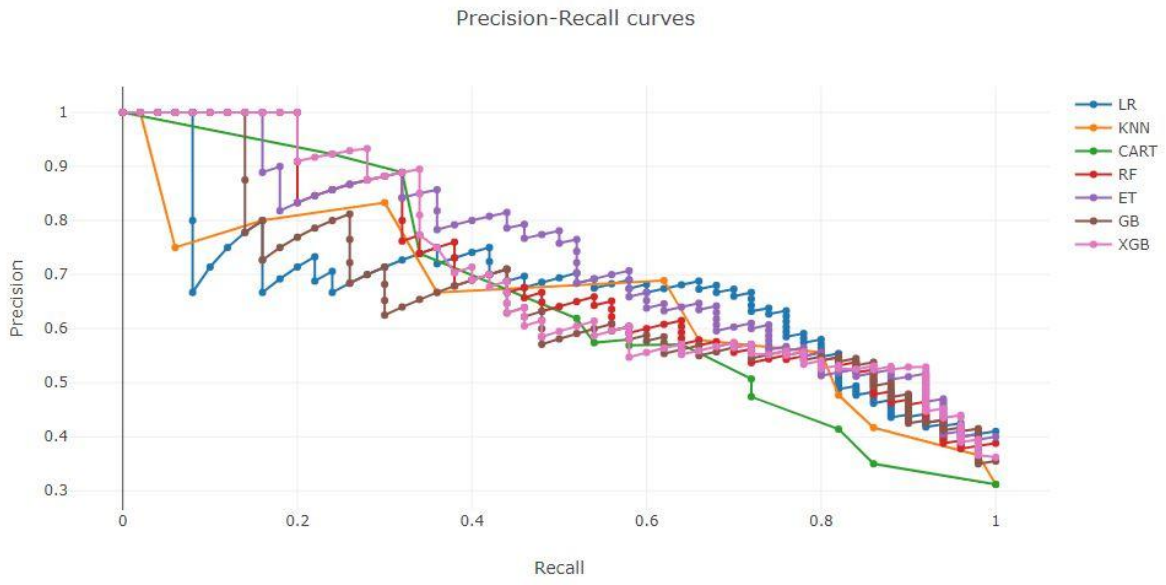
Obrázok 31: ROC krivka a AUC

Z grafu je možné vidieť ROC krivky pre jednotlivé modely. Všetky modely, okrem CART, majú AUC hodnotu veľmi blízku hodnote 0,81. Najvyššiu hodnotu AUC má model extra stromov s hodnotou 0,84. Hranice pravdepodobnosti, ktoré boli zvolené pre jednotlivé body v grafe sa zobrazia po nájdení kurzorom na požadovaný bod (interaktívne grafy v prílohe). Samostatné ROC krivky je možné vidieť kliknutím do legendy grafu a zvolením požadovaných modelov. Z ROC krivky môžeme teda odhadnúť, akú hranicu pravdepodobnosti zvoliť na predikovanie pozitívnej triedy a aký to bude mať dopad na štatistiky modelu.

5.5.2 KRIVKA PRECÍZNOSŤ-SENZITIVITA (ANGLICKY PRECISION-RECALL CURVE)

Jednou z možností porovnania kvality klasifikačného modelu je vykresliť krivku závislosti precíznosti (anglicky precision) na senzitivite (anglicky recall alebo tiež sensitivity). Body tejto krivky tvoria (podobne ako pri ROC krivke) štatistiky vypočítané pre rôzne hranice pravdepodobnosti. V niektorých prípadoch však môže byť použitie tejto krivky výhodnejšie na posúdenie, ktorý model vybrať, ako použitie ROC krivky.

Krivky závislosti precíznosti na senzitivite pre testované modely sú na obrázku číslo 32.



Obrázok 32: Krivky precíznosti a senzitivity

Z grafu vidieť krivky pre jednotlivé modely. Využitie týchto kriviek na zvolenie najvhodnejšieho modelu ilustrujem na nasledujúcom príklade. Ide o rovnaký prípad ako riešim v tejto práci, avšak pre jednoduchosť som zvolil iné počty pacientov. Predstavme si situáciu, že má lekár 200 pacientov, ktorí čakajú na operáciu, ale lekár je schopný vykonať len 20 operácií. Z týchto 200 pacientov je však iba 100 pacientov, ktorí naozaj operáciu potrebujú, avšak lekár nevie identifikovať, ktorí sú to ľudia, vie iba, že ich je 100. Za účelom kategorizácie ľudí na chorých a zdravých sa vytvoria modely a vykreslí sa ROC krivka a Precision-Recall krivka. Predpokladajme, že krivky vyzerajú rovnako ako krivky na obrázku číslo 31 a 32. Keďže lekár vie, že je schopný vykonať iba 20 zákrokov, tak v grafe hľadá model, ktorý ma pri hodnote senzitivity 0,2 najväčšiu hodnotu precíznosti. Vyberá teda XGB model s hodnotou precíznosti

rovnou 1. Inými slovami je teda XGB model schopný identifikovať 20% pacientov z chorých a pritom neklasifikovať žiadneho zdravého pacienta ako chorého. Tým lekár využil svoju kapacitu na maximum a nevykonal zákrok na zdravom pacientovi zbytočne. V prípade, že by však bol schopný vykonať zákrok na 50-tich pacientoch, výhodnejšie by bolo použiť model extra stromov, pretože má pri hodnote senzitivity 0,5 najvyššiu hodnotu precíznosti.

ZÁVER

Táto práca bola zameraná na porovnanie konvenčných a heuristických metód v data miningu používaných na binárnu klasifikáciu. Teoretická časť práce je rozdelená na štyri kapitoly.

Prvá kapitola sa zaoberá modelom logistickej regresie. Kapitola stručne popisuje model, spôsob akým sa učí (fituje), a taktiež ako predísť vytvoreniu príliš zložitého modelu. Model je následne otestovaný na jednoduchej úlohe, kde jasne vidieť oblasti klasifikácie. Ďalej sú spomenuté nevýhody modelu a na jednoduchom príklade sú tieto nevýhody graficky demonštrované.

Druhá kapitola sa zaoberá modelom rozhodovacích stromov. V kapitole je popísané jeho fungovanie a taktiež sú popísané parametre modelu, ktoré treba nastaviť ešte pred trénovaním. Kapitola obsahuje dva rovnaké príklady ako v prípade logistickej regresie, na ktorých demonštruje výhody a nevýhody použitia tohto modelu.

Tretia kapitola sa zaoberá skupinovými modelmi, pričom vysvetľuje výhody ich použitia. Ako reprezentant zo skupinových modelov je vybraný náhodný les, ktorý je detailnejšie popísaný. Výhoda použitia tohto modelu je demonštrovaná graficky na jednoduchom príklade.

Štvrtá kapitola sa zaoberá boostingom. Ako reprezentant tejto skupiny je vybraný XGB (Extreme-Gradient-Boosting) model, ktorý je následne stručne matematicky popísaný.

Praktická časť práce obsahuje analýzu a modelovanie na reálnych medicínskych dátach s použitím programovacieho jazyka Python.

Prvá a druhá kapitola tejto časti sa venuje predstaveniu dát a knižniciam potrebným na analýzu a modelovanie.

V tretej kapitole praktickej časti sa práca venuje exploratívnej dátovej analýze a čisteniu dát. Dáta sú analyzované a vykreslené z rôznych pohľadov pomocou interaktívnych grafov (možné vidieť v prílohe na CD, grafy sa zobrazia len s pripojením na internet). Na základe týchto analýz sú následne odstránené odľahlé hodnoty, ktoré by mohli ovplyvniť kvalitu modelu.

Štvrtá kapitola praktickej časti sa venuje modelovaniu. Na začiatku sú natrénované modely a porovnaná ich presnosť. Porovnanie presnosti je založené na k-násobnej krížovej validácií, ktorá je detailne vysvetlená. Ďalej sú odskúšané dve rôzne transformácie dát za účelom zlepšenia kvality modelov. Za týmto účelom je taktiež detailne popísané ladenie hyperparametrov modelov aj spôsob nájdenia ich optimálnych hodnôt pomocou Grid Search metódy. V tejto kapitole je taktiež zahrnuté zníženie dimenzie dát na základe významnosti prediktorov vypočítaných pomocou náhodného lesa.

Piata kapitola praktickej časti sa venuje vyhodnoteniu kvality modelov na základe ROC krivky a krivky závislosti precíznosti na senzitivite. V oboch prípadoch sú tieto metódy detailne vysvetlené. Na konci kapitoly je uvedený príklad, prečo je vhodné používať rozdielne metódy na vyhodnotenie kvality modelov v závislosti od požadovaného účelu.

POUŽITÁ LITERATÚRA

[1] HASTIE, Trevor, Robert TIBSHIRANI a J. H. FRIEDMAN. *The elements of statistical learning: data mining, inference, and prediction : with 200 full-color illustrations*. New York: Springer, c2001. ISBN 0-387-95284-5.

[2] ZVÁRA, Karel. *Regresní analýza*. Praha: Academia, 1989. ISBN 80-200-0125-5.

[3] ZLÁMAL, Filip. *Logistická regrese v R*. Brno, 2013. Bakalářska práce. Masarykova univerzita, Přírodovědecká fakulta.

[4] ANDĚL, Jiří. *Základy matematické statistiky*. Vyd. 3. Praha: Matfyzpress, 2011. ISBN 978-80-7378-162-0.

[5] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Boston: O'Reilly Media, 2017. ISBN 9781491962299.

[6] FISCHER, R. A. *The use of multiple measurements in taxonomic problems*. Annals of Eugenics. 7 (2): 179–188

[7] DUA, D. and GRAFF, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/iris]. Irvine, CA: University of California, School of Information and Computer Science

[8] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, et al.. *API design for machine learning software: experiences from the scikit-learn project*. European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases, Sep 2013, Prague, Czech Republic. ffhal-00856511

[9] KOMPRDOVÁ, Klára. *Rozhodovací stromy a lesy*. Brno: Akademické nakladatelství CERM, 2012. ISBN 9788072047857.

[10] BREIMAN, Leo, Jerome FRIEDMAN, Charles J. STONE a R.A. OLSHEN. *Classification and Regression Trees*. New York: Chapman and Hall, 1984. ISBN 9780412048418.

[11] RASCHKA, Sebastian a Vahid MIRJALILI. *Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow*. Second edition. Birmingham: Pack publishing, 2017. ISBN 1787125939.

[12] BREIMAN, Leo. Machine Learning (1996) 24: 123. https://doi.org/10.1023/A:1018054314350

[13] FRIEDMAN, Jerome, Trevor HASTIE a Robert TIBSHIRANI. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*. 1998, 1998(28), 337-407.

[14] CHEN,T., GUESTRIN ,C. *XGBoost: A scalable Tree boosting system* . Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York: ACM, 2016. ISBN 978-1-4503-4232-2

[15] KAGGLE. Datasets. kaggle.com [online]. ©2019 [cit. 2019-05-14].
Dostupné z: https://www.kaggle.com/datasets

[16] PYTHON manuál [online]. [cit. 2019-05-14]. Dostupné z: https://www.python.org/doc

[17] JUPYTER NOTEBOOK manuál [online]. [cit. 2019-05-14].
Dostupné z: https://jupyter.org/documentation

[18] PANDAS manuál[online]. [cit. 2019-05-14].
Dostupné z: https://pandas.pydata.org/pandas-docs/stable/

[19] NUMPY manuál[online]. [cit. 2019-05-14].
Dostupné z: https://docs.scipy.org/doc/numpy/dev/

[20] MATPLOTLIB manuál [online]. [cit. 2019-05-14].
Dostupné z: https://matplotlib.org/contents.html

[21] SEABORN manuál [online]. [cit. 2019-05-14].
Dostupné z: https://seaborn.pydata.org/tutorial.html#tutorial

[22] PLOTLY manuál [online]. [cit. 2019-05-14]. Dostupné z: https://plot.ly/python/

[23] SKLEARN manuál [online]. [cit. 2019-05-14].
Dostupné z: https://scikit-learn.org/stable/documentation.html

PRÍLOHY NA CD

Priložené CD obsahuje nasledujúce súbory:

- *diplomova_praca_text.pdf* - text diplomovej práce
- *ilustracne_priklady.html* - zdrojový kód príkladov použitých v teoretickej časti práce
- *prakticka_cast.html* - zdrojový kód + interaktívne grafy použité v praktickej časti