Modeli slučajnih šuma i primjene

Čular, Marko

Modeli slučajnih šuma i primjene

Čular, Marko

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Marko Čular

MODELI SLUČAJNIH ŠUMA I PRIMJENA

Diplomski rad

Voditelj rada:
prof.dr.sc.Siniša Slijepčević

Zagreb, rujan 2020.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

”Svi naši snovi se mogu ostvariti, samo ako imamo dovoljno hrabrosti da idemo za njima.”

Walt Disney

Zahvaljujem se mentoru prof.dr.sc. Siniši Slijepčeviću na ukazanom povjerenju i suradnji prilikom izrade rada.
Veliko HVALA obitelji na bezuvjetnoj ljubavi. Uz vašu podršku i vrijednosti koje ste mi usadili, ne postoje nepremostive prepreke.
Hvala prijateljima na nezaboravnim trenutcima i doživljajima tijekom studiranja. Bez njih bi diplomski rad bio napisan ranije.

Sadržaj

Uvod

Tijekom 21. stoljeća svjetska ekonomija nalazi se u velikom naletu i ekspanziji. Sigurno je da tehnološki napredak ima veliku ulogu u razvoju gospodarstava diljem globusa. U ovom diplomskom radu bavimo se tehnološkom granom koja je još nedovoljno istražena, a to je strojno učenje. Konkretnije, opisujemo model strojnog učenja pod nazivom slučajne šume i njihovu upotrebu u računanju kreditnog rizika.
Slučajne šume su algoritam koji ima razne primjene u svakodnevnici. Njegovom implementacijom možemo dovesti do napretka u nekoliko grana ekonomije, a jedna od njih je bankarstvo. Poboljšanjem učinkovitosti u bankarstvu, to jest u kreditnom sektoru, bavit ćemo se u ovom radu.
Prvo poglavlje posvećeno je teorijskoj obradi modela linearne regresije, kao teoretskog temelja strojnog učenja. Predstavljene su jednostavna i višestruka regresija te metode pomoću kojih dolazimo do temeljnih izračuna.
U drugom poglavlju kroz kratki pregled razvoja strojnog učenja uvodimo čitatelja u tematiku rada. Zatim uvodimo pojam strojnog učenja te opisujemo djelatnosti u kojima se primjenjuje ili postoji potencijal za njegovu primjenu.
U trećem poglavlju bavimo se stablima odluke. Opisani su algoritmi pomoću kojih gradimo stabla te načini na koje granamo čvorove u stablima.
U četvrtom poglavlju nadograđujemo podatke iz trećeg poglavlja na teoriju o slučajnim šumama. Detaljno je napisano kako se algoritam provodi te su prodiskutirane poteškoće koje se mogu javiti za vrijeme provođenja modela.
Konkretnom primjenom slučajnih šuma na računanje rizika kredita u bankama bavimo se u petom poglavlju. Opisan je problem iz svakodnevnog života te se on, pomoću algoritma, pokušava pojednostaviti. Za kraj poglavlja su dobiveni rezultati pojašnjeni kako bi se dobila realna slika o značaju algoritma.
Šesto poglavlje se nadovezuje na peto. U njemu su napisani kodovi za algoritam i izradu grafikona.

Poglavlje 1

Linearna regresija

1.1 Regresijska analiza

Tijekom proučavanja podataka dobivenih kroz mjerenja i istraživanja ponekad je teško uočiti postoji li uzročna veza između danih varijabli. Regresijska analiza je statistička metoda za procjenu odnosa između ovisne varijable (često zvana "varijabla ishoda") koja se označava sa Y_i i jedne ili više neovisnih varijabli x_i (često zvana "prediktorna varijabla"). Pomoću nje dolazimo do matematičke formule kojom opisujemo povezanost podataka. Dakle, cilj svega je na temelju sparenih mjerenja $(x_1, y_1), ..., (x_n, y_n)$ donijeti zaključke o ovisnosti niza slučajnih varijabli Y_i o nezavisnoj varijabli x_i. [13]

Najčešći oblik regresijske analize je linearna regresija u kojoj se pronalazi pravac (ili složenija linearna kombinacija) koji najviše odgovara podacima prema određenom kriteriju. Jedan od najranije upotrebljavanih kriterija je metoda najmanjih kvadrata koju je prvi objavio Carl Friedrich Gauss početkom 19. stoljeća.

Regresijska analiza omogućava procjenu uvjetnog očekivanja ovisne varijable kada neovisne varijable preuzmu određeni skup vrijednosti. Iz tog razloga ona se koristi u dvije različite svrhe. Prvenstveno, upotrebljava se za predviđanje, gdje se njezina primjena značajno preklapa s poljem strojnog učenja. Zatim, kao što je već napisano, u nekim se situacijama regresijska analiza može upotrijebiti za zaključivanje uzročno posljedičnih odnosa između neovisnih i ovisnih varijabli.

Razlikujemo tri tipa regresije: jednostavnu linearnu regresiju, višestruku linearnu regresiju i nelinearnu regresiju. Pobliže ćemo se upoznati s jednostavnom i višestrukom linearnom regresijom.

1.2 Jednostavna linearna regresija

Za početak poglavlja uvedimo definicije vjerojatnosnog prostora i slučajnog uzorka.

Definicija 1.2.1. *Neka su* $\{x_1, x_2, ..., x_n\} \in \Omega$ *slučajne varijable. Familija podskupova* $\mathcal{F}$ *od* Ω *zove se* σ*-algebra (ili* σ*-algebra događaja), ako vrijede sljedeća tri svojstva:*

- $\Omega \in \mathcal{F}$
- *Ako je* $A \in \mathcal{F}$*, onda je i* $A^c \in \mathcal{F}$ *(zatvorenost na komplement)*
- *Ako su* $A_j \in \mathcal{F}$, $j \in \mathbb{N}$*, onda je i* $\bigcup_{j=1}^{\infty} \in \mathcal{F}$ *(zatvorenost na prebrojive unije).*

Uređeni par $(\Omega, \mathcal{F})$ *zove se izmjeriv prostor.*

Definicija 1.2.2. *Neka je* Ω *neprazan skup i* $\mathcal{F}$ σ*-algebra događaja. Vjerojatnost na izmjerivom prostoru* $(\Omega, \mathcal{F})$ *je funkcija* $\mathbb{P} : \mathcal{F} \to [0, 1]$ *koja zadovoljava sljedeća tri aksioma:*

- *(A1) (nenegativnost) Za sve* $A \in \mathcal{F}, \mathbb{P}(A) \geq 0$*;*
- *(A2) (normiranost)* $\mathbb{P}(\Omega) = 1$*;*
- *(A3) (*σ*-aditivnost) Za svaki niz* $(A_j)_{j\in\mathbb{N}}$ *po parovima disjunktnih događaja* $A_j \in \mathcal{F}(A_i \cap A_j = \varnothing$ *za* $i \neq j)$ *vrijedi*

$$\mathbb{P}\left(\bigcup_{j=1}^{\infty} A_j\right) = \sum_{j=1}^{\infty} \mathbb{P}(A_j)$$

Uređena trojka $(\Omega, \mathcal{F}, \mathbb{P})$ *zove se vjerojatnosni prostor. [14]*

Definicija 1.2.3. *Neka je X slučajna varijabla s funkcijom distribucije* $F(x)$*. Slučajni uzorak veličine n za slučajnu varijablu X je slučajni vektor* $(X_1, X_2, ..., X_n)$*, gdje su sve slučajne varijable* X_i*,* $i = 1, ..., n$*, nezavisne sa zajedničkom funkcijom distribucije vjerojatnosti* $F(x)$*.[16]*

Nakon što smo definirali pojmove na kojima ćemo temeljiti daljnju teoriju, možemo krenuti s teorijskim pregledom jednostavne (ili jednostruke) linearne regresije.
Kao što samo ime sugerira, jednostavna linearna regresija je model s jednom varijablom. Tiče se dvodimenzionalnih točaka s jednom neovisnom i jednom ovisnom varijablom koje se uobičajeno prikazuju u Kartazijevu koordinatnu sustavu. Tendencija je pronaći linearnu

funkciju (pravac) koja predviđa vrijednosti ovisne varijable kao funkciju neovisnih varijabli. Riječ jednostavna odnosi se na činjenicu da je varijabla ishoda povezana s jednom prediktornom varijablom.[6] Jednostavni regresijski model zapisujemo na sljedeći način:

$$y_i = ax_i + b + \epsilon_i \tag{1.1}$$

Ovdje su:

- y_i vrijednosti zavisne varijable ovisne o vrijednostima prediktorne varijable x_i
- a i b nepoznati parametri koje tek treba odrediti na temelju dostupnih podataka pomoću metode najmanjih kvadrata
  - a predstavlja nagib regresijske jednadžbe
  - b se naziva konstantni član jednadžbe te predstavlja odsječak na y-osi (očekivanu vrijednost Y kada varijabla x iznosi 0)
- $x_1, x_2, ..., x_n$ vrijednosti prediktorne varijable x koja se proučava
- $\epsilon_1, \epsilon_2, ..., \epsilon_n$ slučajne greške (reziduali), tj. razlika između očekivane vrijednosti i postignute vrijednosti varijable y_i.

Veza između osnovnih parametara a i b i točaka (x_i, y_i) naziva se modelom linearne regresije. Definirajmo dva pojma koja ćemo koristiti u sljedećem ulomku (definicije pronađene u izvoru [16]).

Definicija 1.2.4. *Procjenitelj nepoznatog parametra t je funkcija slučajnog uzorka $\hat{T} = h(x_1, x_2, ..., x_n)$.*

Definicija 1.2.5. *Procjenitelj $\hat{T}$ je nepristran za parametar t ako je očekivanje od $\hat{T}$ jednako vrijednosti parametra t:*

$$E(\hat{T}) = t.$$

Cilj je pronaći "dobre" procjene vrijednosti $\hat{a}$, $\hat{b}$ i $(\hat{\sigma})^2$ za nepoznate parametre a, b i σ, gdje je σ oznaka za standardnu devijaciju.

Gauss-Markovljevi uvjeti

Potrebno je utvrditi pretpostavke pod kojima će metoda najmanjih kvadrata najbolje raditi, to jest biti što točnija. Ima ih četiri i nazivaju se Gauss-Markovljevi uvjeti. Trebamo biti svjesni kako oni neće uvijek biti u potpunosti zadovoljeni, budući da u stvarnosti nemamo uvijek idealno raspoređene podatke. Međutim, model linearne regresije s idealnim uvjetima može se koristiti kao reprezentativni primjer za usporedbu s drugim "realnijim" modelima.

Definicija 1.2.6. *Gauss-Markovljevi uvjeti za slučajne greške* ϵ_i, $i = 1, ..., n$ *su:*

1. $E\{\epsilon_i\} = 0$, $i = 1, ..., n$
2. $\epsilon_i \sim N(0, \sigma^2)$
3. $Cov\{\epsilon_i, \epsilon_j\} = 0$ *za* $i, j = 1, ..., n$, $i \neq j$
4. $Var\{\epsilon_i\} = \sigma^2$, $i = 1, ..., n$.

Pojasnimo dane uvjete.
1. Očekivana greška iznosi 0, što znači da je veza između prediktorne i izlazne varijable doista linearna.
2. Greške su normalno distribuirane.
3. Korelacija između grešaka varijabli iznosi nula, to jest promatrane varijable ne utječu međusobno jedna na drugu.
4. Raspršenost grešaka je konstantna.

Ako su navedeni uvjeti ispunjeni, tada imamo najbolju linearnu nepristranu procijenu (eng. *BLUE - best linear unbiased estimate*). Najbolju, jer je varijanca minimalna, to jest manja nego za bilo koje druge procjenitelje. Zatim linearnu, jer to svojstvo slijedi iz prve pretpostavke, u suprotnom ne možemo primjeniti metodu najmanjih kvadrata. Naposljetku, nepristranu, jer su očekivane vrijednosti α i β uistinu koeficijenti koji najbolje opisuju vezu između ulazne i izlazne varijable.
Sada, kada znamo nužne uvjete, možemo prijeći na metodu najmanjih kvadrata.

Metoda najmanjih kvadrata

U statistici postoje razne metode za procjenu rezultata. Vjerojatno najjednostavnija je aritmetička sredina.

Definicija 1.2.7. *Za dane vrijednosti $x_1, ..., x_N$ definiramo aritmetičku sredinu kao*

$$\frac{(x_1 + ... + x_N)}{N}. \tag{1.2}$$

Aritmetičku sredinu označavamo sa

$$\overline{x} = \frac{1}{N}\sum_{n=1}^{N} x_n. \tag{1.3}$$

Na primjer, imamo dva niza podataka: $\{5, 15, 25, 35, 45, 55\}$ i $\{30, 30, 30, 30, 30, 30\}$. Oba skupa podataka imaju istu aritmetičku sredinu, ali prvi skup ima veće oscilacije između članova dok se drugi sastoji od niza istih brojeva. To nas navodi na proučavanje varijance, kako bi utvrdili koliko članovi niza odstupaju od srednje vrijednosti.

Definicija 1.2.8. *Varijanca niza podataka $\{x_1, ..., x_N\}$ označava se sa σ_x^2, a računa se pomoću formule:*

$$\sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}(x_i - \overline{x})^2 \tag{1.4}$$

Definicija 1.2.9. *Standardna devijacija je kvadratni korijen varijance:*

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_i - \overline{x})^2} \tag{1.5}$$

te pomoću nje dolazimo do razlike u vrijednostima između danih vrijednosti i njihove aritmetičke sredine [9].

Kao što vidimo na slici 1.1, postoji nekoliko pozicija linija koje se čine približno dobre. Stoga je potrebno odrediti pravac koji je najbliži svim podacima. Jednadžba općenitog pravca glasi $y = ax + b$ te tada izraz $y - (ax + b)$ iznosi nula. Prema tome

$$\{(x_1, y_1), ..., (x_N, y_N)\}$$

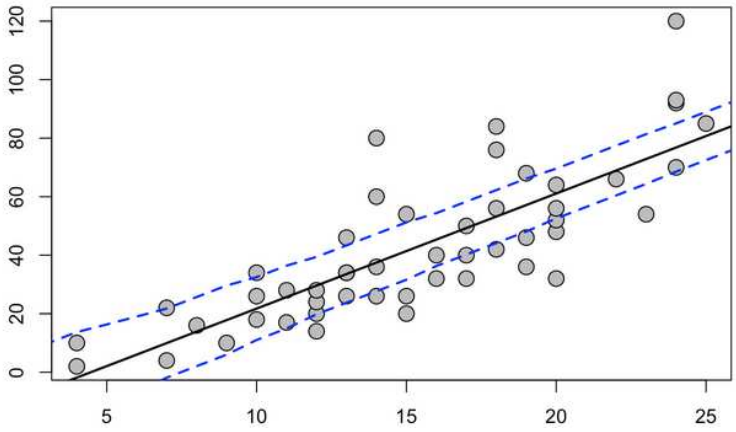
možemo zapisati kao

$$\{y_1 - (ax_1 + b), ..., y_N - (ax_N + b)\} \tag{1.6}$$

Varijanca za, na taj način zapisane podatke, glasi

$$\sigma^2_{y-(ax+b)} = \frac{1}{N}\sum_{n=1}^{N}(y_n - (ax_n + b))^2 \tag{1.7}$$

Veliki iznosi posebno dobivaju na težini zbog kvadriranja. Stoga ovaj pristup favorizira mnoge pogreške "srednje veličine" nad nekolicinom pogrešaka velikog raspona.



Slika 1.1: Regresijski pravac

Definicija 1.2.10. *Za dani skup podataka* $\{(x_1, y_1), ..., (x_N, y_N)\}$*, definiramo grešku kao*

$$E(a, b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2 \tag{1.8}$$

Cilj metode je pronaći procijenitelje a i b takve da je pogreška, odnosno suma kvadrata razlike između prave i predviđene varijable, minimalna.
Do njih ćemo doći rješavanjem jednadžbi

$$\frac{\partial E}{\partial a} = 0, \frac{\partial E}{\partial b} = 0 \tag{1.9}$$

Deriviranjem dobivamo:

$$\frac{\partial E}{\partial a} = \sum_{n=1}^{N} 2(y_n - (ax_n + b)) \cdot (-x_n), \tag{1.10}$$

$$\frac{\partial E}{\partial b} = \sum_{n=1}^{N} 2(y_n - (ax_n + b)) \cdot 1 \tag{1.11}$$

Izjednačimo $\frac{\partial E}{\partial a} = \frac{\partial E}{\partial b} = 0$:

$$\sum_{n=1}^{N} (y_n - (ax_n + b)) \cdot x_n = 0 \tag{1.12}$$

$$\sum_{n=1}^{N} (y_n - (ax_n + b)) = 0 \tag{1.13}$$

Jednadžbe možemo zapisati kao

$$\left(\sum_{n=1}^{N} x_n^2\right) a + \left(\sum_{n=1}^{N} x_n\right) b = \sum_{n=1}^{N} x_n y_n \tag{1.14}$$

$$\left(\sum_{n=1}^{N} x_n\right) a + \left(\sum_{n=1}^{N} 1\right) b = \sum_{n=1}^{N} y_n \tag{1.15}$$

Dobivamo da vrijednosti a i b koje minimiziraju grešku zadovoljavaju sljedeću matričnu jednadžbu:

$$\begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{pmatrix} \tag{1.16}$$

Pokazat ćemo da je matrica invertibilna, što implicira

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{pmatrix} \tag{1.17}$$

Označimo matricu s M. Determinanta od M je

$$\det M = \sum_{n=1}^{N} x_n^2 \cdot \sum_{n=1}^{N} 1 - \sum_{n=1}^{N} x_n \cdot \sum_{n=1}^{N} x_n. \tag{1.18}$$

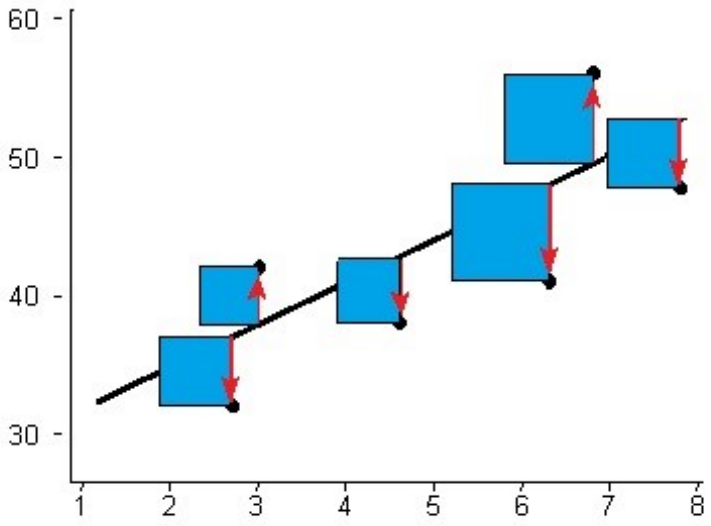
Iz 1.3 dobivamo

$$\det M = N \sum_{n=1}^{N} {x_n}^2 - (N\overline{x})^2 \tag{1.19}$$

$$= N^2 \cdot \left(\frac{1}{N} \sum_{n=1}^{N} x_n^2 - \overline{x}^2\right) \tag{1.20}$$

$$= N^2 \cdot \frac{1}{N} \sum_{n=1}^{N} (x_n - \overline{x})^2. \tag{1.21}$$

Prema tome, sve dok svi x_n nisu jednaki, det M će biti različita od nule i M će biti invertibilna. Stoga zaključujemo, sve dok x_n nisu svi jednaki, najbolje vrijednosti parametara a i b dobivamo rješavanjem linearnog sustava jednadžbi; rješenje je dano u 1.17. Također se možemo zapitati zašto smo odabrali metodu najmanjih kvadrata. Funkcija 1.8 koju minimiziramo je diferencijabilna, stoga je rješenje jedinstveno i može se zapisati u zatvorenoj formi te je to razlog zašto koristimo najmanje kvadrate. Na primjer, koristeći metodu koja promatra sumu apsolutnih vrijednosti reziduala nemamo neprekidno derivabilnu funkciju. Ono što dodatno pridonosi popularnosti metode najmanjih kvadrata su Gauss-Markovljevi uvjeti navedeni u prošlom poglavlju jer se njima potvrđuje učinkovitost metode.



Slika 1.2: Prikaz metode najmanjih kvadrata i prilagodbe regresijskog pravca

Rasipanje podataka

Tijekom istraživanja i analiziranja razvijeni su razni pristupi i interpretacije modela radi jednostavnijeg i boljeg razumijevanja podataka. U poglavlju 1.2 smo ϵ_i definirali kao slučajne pogreške, tj. razlike između očekivane vrijednosti varijable Y_i i postignute vrijednosti. U matematičkom zapisu prethodna rečenica nam daje:

$$\hat{\epsilon}_i = y_i - \hat{\mu}_i, \tag{1.22}$$

gdje je $\hat{\mu}_i = \hat{a}x_i + \hat{b}$, a $(x_1, y_1), ..., (x_n, y_n)$ niz mjerenja. Veličinu $\hat{\epsilon}_i$ nazivamo rezidual.
Pauše navodi kako se dio $\hat{\epsilon}_i$ izlazne varijable $y_i = \hat{\mu}_i + \hat{\epsilon}_i$ ne može objasniti funkcijskom ovisnošću izlaza o ulazu, već potječe od djelovanja slučajnih faktora (slučajne greške ϵ_i) (vidi [13]).

Definicija 1.2.11. *Definirajmo korigiranu varijancu niza podataka kao*

$$s^2 = \frac{1}{n-2}\sum_{i=1}^{n}\left(y_i - \hat{a}x_i - \hat{b}\right)^2 = \frac{n}{n-2}\left({s_y}^2 - \frac{{s_{xy}}^2}{{s_x}^2}\right), \tag{1.23}$$

gdje su

$${s_y}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2\,, \tag{1.24}$$

$$s_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}), \tag{1.25}$$

$$s_x{}^2 = \frac{1}{n}\sum_{i=1} n(x_i - \overline{x})^2. \tag{1.26}$$

Iz 1.23 i 1.7 može se zapisati

$$s_y{}^2 = \hat{\sigma}^2 + \frac{s_{xy}{}^2}{s_x{}^2} \tag{1.27}$$

Veličina $\hat{\sigma}^2$ opisuje rasipanje izlaznih podataka oko procjenjene regresijske funkcije, dok $s_y{}^2$ opisuje rasipanje izlaznih podataka oko njihove sredine $\overline{y}$.
Lako se dokaže da je

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{\mu}_i - \overline{y})^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{a}x_i + \hat{b} - \overline{y})^2 = \frac{s_{xy}{}^2}{s_x{}^2}, \tag{1.28}$$

pa se vidi da se veličina

$$\hat{\sigma_0}{}^2 = \frac{s_{xy}{}^2}{s_x{}^2} \tag{1.29}$$

može interpretirati kao mjera rasipanja vrijednosti procijenjene regresijske funkcije (prognoziranih vrijednosti izlaza) oko $\overline{y}$. U tom se svjetlu relacija 1.29, zapisana kao

$$s_y{}^2 = \hat{\sigma}^2 + \hat{\sigma}_0^2, \tag{1.30}$$

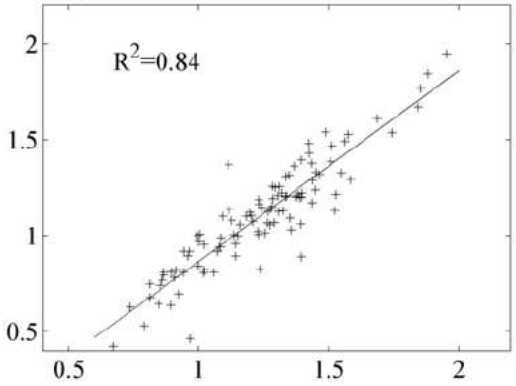
može interpretirati tako da se kaže da je rasipanje izlaznih podataka oko njihove aritmetičke sredine jednako zbroju rasipanja uzrokovanog regresijskom ovisnošću (funckijom $x_i \mapsto \hat{\mu}_i$) i rasipanja uzrokovanog slučajnom greškom, tzv. *rezidualnog rasipanja*. Veličina

$$R^2 = \frac{\hat{\sigma}_0^2}{s_y{}^2} = 1 - \frac{\hat{\sigma}^2}{s_y{}^2} \tag{1.31}$$

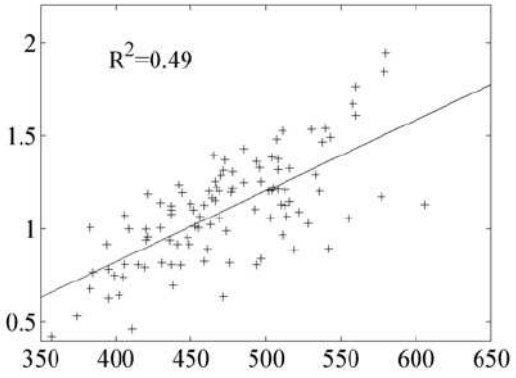
zove se *koeficijent determinacije*. [13]
Iz formule se lako vidi da vrijedi

$$0 \leq R^2 \leq 1. \tag{1.32}$$

Možemo se zapitati uz koje uvjete se postižu minimalne i maksimalne vrijednosti koeficijenta determinacije, odnosno 0 i 1. Iz 1.31 se vidi da će R iznositi nula ako je $\hat{\sigma}_0^2 = 0$ ili $\hat{\sigma}^2 = s_y{}^2$. U tom slučaju regresijski pravac je paralelan s osi apscisa iz čega možemo izvesti zaključak da nezavisna (ulazna) varijabla x ne utječe na izlaznu slučajnu varijablu



Slika 1.3: Visoka vrijednost korelacije



Slika 1.4: Srednja vrijednost korelacije

Y. U tom slučaju rasipanje podataka ne možemo objasniti i prikazati funkcijskom vezom između x i Y. Pauše tvrdi da nam tada poznavanje ulazne vrijednosti x ne omogućuje da se bilo što novo kaže o pripadnom izlazu y_x, što već ne bi bilo moguće reći i bez poznavanja x.[13]

Drugi ekstrem je $R = 1$, a postiže se kada je $\hat{\sigma}^2 = 0$. U tom slučaju svi podaci leže točno na regresijskoj liniji te tada nema rezidualnog rasipanja.

Iako je visoki koeficijent determinacije poželjan, ne postoje egzaktni podaci koliki on treba biti kako bi regresija bila uspješna. Niska vrijednost koeficijenta ne mora uvijek predstavljati negativnost. Ako imamo statistički značajne prediktore, možemo izvesti zaključke o tome kako su promjene u vrijednostima prediktora povezane s promjenama vrijednosti u izlaznim varijablama. Bez obzira na R^2, koeficijenti predstavljaju srednju promjenu u odgovoru za jednu jedinicu prediktora dok su drugi prediktori u modelu konstantni, stoga ova vrsta infomacija može biti od koristi. Također, budući da koeficijent determinacije ovisi o varijanci grešaka, s njenim povećanjem povećava se i udio varijance izlazne varijable koji je neobjašnjiv pomoću regresije. Samim time je i koeficijent determinacije manji, iako regresijska krivulja, grafički gledano, dobro opisuje očekivanje izlazne varijable.

Znači li to da je visoki koeficijent determinacije uvijek dobar? Ne, visoki R^2 ne znači nužno da se model dobro uklapa. Često su moguća dodatna poboljšanja modela kako bi on bio što prikladniji.

1.3 Višestruka linearna regresija

U prošlom poglavlju u regresijskom modelu je izlazna varijabla Y ovisila samo o jednoj ulaznoj (nezavisnoj) varijabli x. U ovom poglavlju napravit ćemo pomak te proučavati model višedimenzionalne regresije. Svrha višedimenzionalne linearne regresije je modelirati vezu izmedu izlazne (zavisne) varijable y i dvije ili više prediktornih (nezavisnih) varijabli x_k.
Opća formula višedimenzionalne linearne regresije glasi:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon. \tag{1.33}$$

Kao i kod jednostavne linearne regresije, y je izlazna varijabla, $\beta_0, ..., \beta_k$ su nepoznati regresijski koeficijenti koji se određuju pomoću metode najmanjih kvadrata, k je broj prediktornih varijabli, a ϵ predstavlja razliku između očekivane vrijednosi varijable Y i postignute vrijednosti (skraćeno: slučajne greške). Formulu 1.33 možemo zapisati kao

$$y = X\beta + \epsilon \tag{1.34}$$

gdje je

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_n \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ \beta_k \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 1 & x_1 2 & .. & x_1 k \\ 1 & x_2 1 & x_2 2 & .. & x_2 k \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & x_n 1 & x_n 2 & .. & x_n k \end{bmatrix}, \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ . \\ . \\ \epsilon_n \end{bmatrix}$$

Gauss-Markovljevi uvjeti

Da bi se koeficijenti regresije procijenili precizno, nužno je da sljedeći uvjeti budu zadovoljeni. Ne budu li zadovoljeni, primjena metode najmanjih kvadrata može rezultirati znatno različitim koeficijentima β ovisno o tome na kojem skupu podataka ih primjenjujemo.
Iako su Gauss-Markovljevi uvjeti već definirani u poglavlju 1.2, napisat ćemo ih ponovno.

1. $\mathrm{E}\{\epsilon_i\} = 0, i = 1, ..., n$
2. $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$
3. $\mathrm{Cov}\{\epsilon_i, \epsilon_j\} = 0$ za $i, j = 1, ..., n, i \neq j$
4. $\mathrm{Var}\{\epsilon_i\} = \sigma^2, i = 1, ..., n.$

U istom smo poglavlju naveli kako uz zadovoljene uvjete imamo najbolju linearnu nepristranu procijenu (*eng. BLUE*). Tu tvrdnju potkrjepljujemo idućim teoremom.

Teorem 1.3.1 (Gauss-Markovljev teorem)**.** *Neka vrijede Gauss-Markovljevi uvjeti i neka je procjenitelj $\hat{\beta}$ dobiven metodom najmanjih kvadrata. Tada je $\hat{\beta}$ najbolji linearni nepristrani procjenitelj.*

Dokaz teorema slijedi u sljedećem podpoglavlju.

Metoda najmanjih kvadrata

Za procjenu koeficijenata regresije ćemo, kao i kod jednostavne linearne regresije, koristiti metodu najmanjih kvadrata. Cilj je izračunati vektor najmanjih kvadrata pomoću minimiziranja zbroja kvadrata reziduala [11]:

$$S(\beta) = \sum_{i=1}^{n} \epsilon_i^2 = \epsilon'\epsilon = (y - X\beta)'(y - X\beta) \tag{1.35}$$

Minimizirana suma kvadrata dobiva se deriviranjem i izjednačavanjem s nulom:

$$\frac{\partial S}{\partial \beta}|_{\hat{\beta}} = -2X'y + 2X'X\hat{\beta} = 0 \tag{1.36}$$

Pojednostavimo:

$$X'X\hat{\beta} = X'y \tag{1.37}$$

Pomnoženo s inverzom $(X'X)^{-1}$ dobivamo:

$$\hat{\beta} = (X'X)^{-1}X'y \tag{1.38}$$

Pod uvjetom da inverz postoji, to jest da su prediktori linearno nezavisni, regresijski model koji odgovara promatranim vrijednostima je:

$$\hat{y} = X\hat{\beta} = X(X'X)^{-1}X'y. \tag{1.39}$$

Ako greške imaju očekivanu vrijednost nula, jednaku varijancu i korelacija između njih iznosi nula, tada je, prema teoremu 1.3.1, $\hat{\beta}$ najbolja linearna nepristrana procijena.
Sada, kada znamo koliko iznosi $\hat{\beta}$, možemo dokazati teorem 1.3.1.

Dokaz teorema 1.3.1. Neka je $\hat{\beta} = (X'X)^{-1}X'$ dobivena metodom najmanjih kvadrata. Neka je $\tilde{\beta} = Cy$ drugi linearni procjenitelj od β, gdje je $C = (X'X)^{-1}X' + D$ te je D $K \times n$ matrica. Cilj je dokazati kako takav procjenitelj nema varijancu manju od od $\hat{\beta}$. Budući da

je Cy nepristran, računamo:

$$\begin{aligned}
E[\tilde{\beta}] &= E[Cy] \\
&= E[((X'X)^{-1}X' + D)(X\beta + \epsilon)] \\
&= ((X'X)^{-1}X' + D)X\beta + ((X'X)^{-1}X' + D)E[\epsilon] \quad \text{(prema prvoj pretpostavci je } E[\epsilon] = 0) \\
&= ((X'X)^{-1}X' + D)X\beta \\
&= (X'X)^{-1}X'X\beta + DX\beta \\
&= (I_K + DX)\beta
\end{aligned}$$

Slijedi da je $\tilde{\beta}$ nepristran ako i samo ako je $DX = 0$. Tada je:

$$\begin{aligned}
\mathrm{Var}(\tilde{\beta}) &= \mathrm{Var}(Cy) \\
&= C\ \mathrm{Var}(y)C' \\
&= \sigma^2 CC' \\
&= \sigma^2((X'X)^{-1}X' + D)(X(X'X)^{-1} + D') \\
&= \sigma^2((X'X)^{-1}X'X(X'X)^{-1} + (X'X)^{-1}X'D' + DX(X'X)^{-1} + DD') \\
&= \sigma^2(X'X)^{-1} + \sigma^2(X'X)^{-1}(DX)' + \sigma^2 DX(X'X)^{-1} + \sigma^2 DD' \\
&= \sigma^2(X'X)^{-1} + \sigma^2 DD' && [DX = 0] \\
&= \mathrm{Var}(\hat{\beta}) + \sigma^2 DD' && [\sigma^2(X'X)^{-1} = \mathrm{Var}(\hat{\beta})]
\end{aligned}$$

Budući da je DD' pozitivna semi-definitna matrica, tvrdnja teorema je dokazana. □

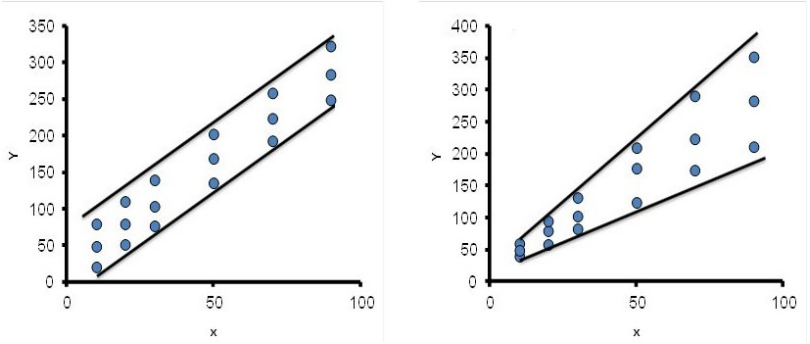
Moguće pogreške

U ovom poglavlju opisane su moguće pogreške koje se mogu dogoditi prilikom korištenja modela višedimenzionalne linearne regresije.

Heteroskedastičnost

Jedan od Gauss-Markovljevih uvjeta je da reziduali imaju konstantnu varijancu. Kada ta pretpostavka klasičnog linearnog regresijskog modela nije poštivana, varijanca odstupanja je promjenjiva, tj. zavisi o opažanju i, tj.

$$\mathrm{Var}(\epsilon_i) = \sigma_i^2$$



Slika 1.5: Na lijevoj slici je varijanca konstantna te su odstupanja homoskedastična, dok je na desnoj slici vidljivo da je varijanca nekonstantna te su odstupanja heteroskedastična

tada kažemo da su odstupanja heteroskedastična. Ukoliko je ova varijanca stalna, ona ne ovisi o opažanju i, tj.

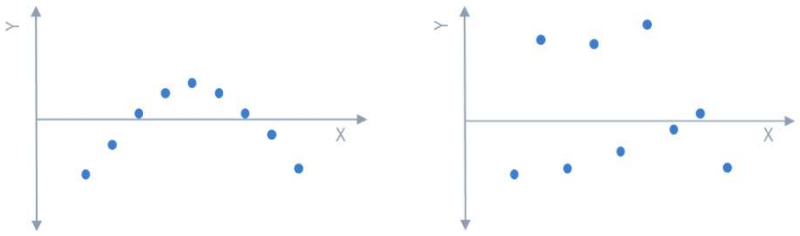
$$\text{Var}(\epsilon_i) = \sigma^2$$

tada kažemo da su odstupanja homoskedstična.[12] Heteroskedastičnost može uzrokovati nekonzistenost standardne devijacije i neefikasnost ocjena parametara pa više nemamo minimalnu varijancu (više nije najbolja linearna nepristrana procjena).

Na slici 1.5 su prikazani primjeri heteroskedastičnosti i homoskedstičnosti. Na desnom grafu (hetero) se jasno vidi nepostojanje konstantne varijance. Uklanjanje heteroskedastičnosti je moguće na dva načina, ovisno je li varijanca poznata ili nije. Kada je varijanca poznata koristi se vagana metoda najmanjih kvadrata (*eng. WLS - weighted least squares*). Ako varijanca nije poznata, što je puno češći slučaj, koriste se transformacije stabiliziranja varijance.

Autokorelacija

Jedan od Gauss-Markovljevih uvjeta je da su $\epsilon_1, ..., \epsilon_n$ nezavisne slučajne varijable. Autokorelacija se javlja kada reziduali nisu međusobno nezavisni, to jest kada slučajne greške pokazuju prepoznatljiv obrazac kretnje. Pojednostavljeno, to je pojava za koju vrijedi $Cov(\epsilon_i, \epsilon_j) \neq 0, i \neq j$. Postojanje autokorelacije može se provjeriti na više načina, grafičkom metodom ili testiranjem.
Grafički prikazujemo reziduale u odnosu na određeni vremenski period (1.6). Ako slučajne greške periodično mijenjaju predznak tada imamo pozitivnu autokorelaciju. U slučaju da reziduali alterniraju, tada se radi o negativnoj autokorelaciji. Jedan od testova za provjeru postojanja autokorelacije je Durbin Watson test. Neka je model u kojem se javlja autoko-



Slika 1.6: Primjeri pozitivne i negativne autokorelacije

relacija (izabrana je jednostruka linearna regresija radi jednostavnijeg prikaza):

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t, \qquad \epsilon_t = \rho\epsilon_{t-1} + v_t$$

Hipoteze Dubrin Watsonovog testa su:

$$d = \frac{\sum_{t=2}^{T}(\epsilon_t - \epsilon_{t-1})^2}{\sum_{t=1}^{T}\epsilon_t^2} \tag{1.40}$$

gdje su ϵ_t reziduali modela, a ρ ocijena autokorelacijskog koeficijenta [11]. Test rezultira brojem između 0 i 4 koji ima sljedeća značenja:

- 0 do 2 - pozitivna autokorelacija
- 2 do 4 - negativna autokorelacija
- 2 - nema autokorelacije

U literaturi se upotrebljava *pravilo palca* koje tvrdi da je raspon od 1.5 do 2.5 normalan, dok Field [3] tvrdi da su vrijednosti manje od 1 i veće od 3 razlog za brigu.

Multikolinearnost

Multikolinearnost je situacija u kojoj su dvije ili više ulaznih varijabli međusobno linearno korelirane, pa je jednu ulaznu varijablu moguće vrlo točno predvidjeti kao linearnu kombinaciju drugih ulaznih varijabli. [18]

Definicija 1.3.2. *Varijable $x_1, ..., x_m$ su multikolinearne ako*

$$w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_m x_m^{(i)} + \epsilon_i = 0 \tag{1.41}$$

za neke vrijednosti $w_0, ..., w_m$ i uz neku malu varijancu odstupanja ϵ_i.

Efekti multikolinearnosti su loša procjena koeficijenata i velika standardna pogreška. Najlakši način rješavanja multikolinearnosti je odstranjivanje jedne ili više prediktorne varijable. Za otkrivanje multikolinearnosti koristi se faktor utjecaja varijance (*eng. VIF - variance inflation factor*). Računa se pomoću formule

$$\mathrm{VIF}_i = \frac{1}{1 - {R_i}^2}, \tag{1.42}$$

gdje je R_i koeficijent determinacije. Ako je VIF_i veći od 5, multikolinearnost je velika.

Mjere utjecaja

Modeli često sadrže outlierse, to jest opažanja koja značajno odstupaju od ostalih opažanja. Iznimno ih je važno detektirati jer mogu imati veliki utjecaj na metodu najmanjih kvadrata, a samim time i na utvrđivanje koeficijenata regresije. Takva točka nije konzistentna s predviđenom vrijednošću. Stoga je nužno ispitati utjecaje na model i ponekad ih ukloniti iz modela. Jedna od metoda koja se koristi u te svrhe je Cookova mjera udaljenosti (*eng. Cook's distance value*). Pomoću te metode mjerimo utjecaje na metodu najmanjih kvadrata kada se izbriše opažanje i. Velike vrijednosti D_i (rezultati iznad 1) ukazuju da i-to opažanje ima značajan utjecaj na metodu najmanjih kvadrata i da ga treba ukloniti. [11]
Cookova udaljenost D_i opažanja i se definira kao zbroj svih promjena u regresijskom modelu kada se opažanje i izuzme iz njega i računa se pomoću sljedeće formule:

$$D_i = \frac{\sum_{j=1}^{n} (\hat{y}_j - \hat{y}_{j(i)})^2}{(p+1)\hat{\sigma}^2}, \tag{1.43}$$

gdje je y_j j-ta izlazna varijabla, $y_{j(i)}$ j-ta izlazna varijabla koja ne uključuje opažanje i, p broj regresijskih koeficijenata i σ varijanca temeljena na svim podacima.

Prikladnost modela

Ispitivanje značajnosti regresije

Nakon što je procjena vrijednosti koeficijenata β završena, važno je znati koliko su rezultati značajni i može li se statistički odrediti linearna veza između prediktora i izlazne varijable. Koriste se F-test i odgovarajuće p vrijednosti kako bi se uvidjelo može li se nul hipoteza odbaciti i samim time potvrditi linearnost. Nul hipoteza je definirana kao:

$$\mathrm{H}_0 : \beta_0 = \beta_1 = ... = \beta_k = 0,$$
$$\mathrm{H}_1 : \beta_j \neq 0, \text{ za barem jedan } j$$

Po definiciji računamo:

$$F = \frac{n-(p+1)}{p} \cdot \frac{RSS_{\text{mali}} - RSS_{\text{veliki}}}{RSS_{\text{veliki}}} \sim F_{p,n-(p+1)}. \tag{1.44}$$

Dakle, u ovom slučaju testiramo je li najjednostavniji model $Y = \beta_0 + \epsilon$ jednak danom modelu pri čemu RSS_{mali} predstavlja raspršenost podataka od prilagođenih vrijednosti osnovnog modela, a RSS_{veliki} danog modela. Ako jest, znači da su koeficijenti β_i (za $i = 1, ..., p$) jednaki nuli, to jest beznačajni, te vrijedi nul hipoteza.

Analiza reziduala

Analiza reziduala ili slučajnih grešaka provodi se kako bi se provjerilo dolazi li do kršanja Gauss-Markovljevih uvjeta. Reziduali su definirani kao odstupanja između očekivane i prave vrijednosti izlazne varijable:

$$\epsilon_i = y_i - \hat{y}_i \tag{1.45}$$

Važno svojstvo reziduala jest da njihova aritmetička sredina iznosi nula i da je prosječna varijanca reziduala definirana kao:

$$\frac{\sum_{i=1}^{n}(\epsilon_i - \overline{\epsilon})^2}{n-p} = \frac{\sum_{i=1}^{n}\epsilon_i{}^2}{n-p} = \frac{SS_{Res}}{n-p} = MS_{Res} \tag{1.46}$$

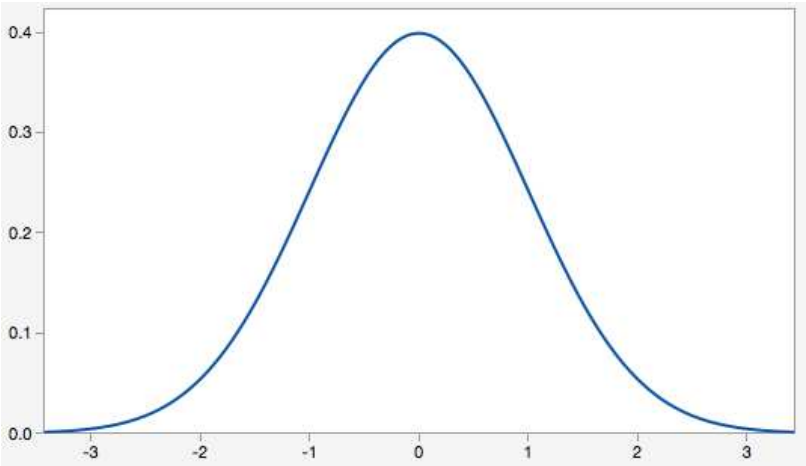
Pomoću skaliranja reziduala lakše se pronalaze outliersi ili ekstremi. Ako rezidual ima vrijednost d_i veću od tri, to je pokazatelj da se radi o outlieru ili ekstremu koje je potrebno procijeniti. [11]

$$d_i = \frac{\epsilon_i}{\sqrt{MS_{Res}}}, i = 1, 2, ..., n. \tag{1.47}$$

Za provjeru normalnosti pretpostavke koristi se graf normalne distribucije (slika 1.7. Mala odstupanja ne utječu mnogo na model. Pomoću grafa reziduala i prikladnih vrijednosti provjeravamo nelinearne veze između varijabli i testiramo model na heteroskedastičnost. Scale-location plot(?) ilustrira je li treća pretpostavka zadovoljena, na primjer imaju li slučajne greške istu varijancu. Reziduali vs leverage graf(?) ilustrira javljaju li se među podacima outliersi koje treba obraditi.

Koeficijent determinacije

U dijelu rada koji se odnosi na jednostavnu regresiju, koeficijent determinacije opisali smo kao mjeru koliko je rasipanja izlaznih podataka uzrokovano regresijskom ovisnošću, a koliko rasipanja čini rezidualno rasipanje. Ideja ostaje ista i kod višestruke linearne regresije.



Slika 1.7: Krivulja normalne distribucije

Dakle, R^2 je vrijednost koja ukazuje koliko je dobro regresijski model prilagođen skupu podataka. R^2 vrijednost je broj između 0 i 1 i definiran je kao:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}. \tag{1.48}$$

Glavni problem koji se javlja prilikom ocjenjivanja modela na temelju koeficijenta determinacije je taj što se dodavanjem druge prediktorne varijable R^2 vrijednost uvijek povećava, a suma kvadrata reziduala smanjuje. To može dovesti do zaključka da je model s više varijabli bolji od onog s manje varijabli, no time može doći do pretjerane specijalizacije. Taj problem može se eliminirati uvođenjem prilagođenog koeficijenta determinacije:

$$\text{adj}R^2 = 1 - \frac{n-1}{n-p-1}(1 - R^2). \tag{1.49}$$

U pravilu je prilagođeni R^2 češće korišten jer sankcionira složenije modele (za veći p je adjR^2 manji). Također, dobro funkcionira i sa malim brojem prediktora.

Poglavlje 2

Strojno učenje

2.1 Povijest strojnog učenja

Kao uvod u materiju napravljen je kratak pregled razvoja strojnog učenja kako bi ispitali porijeklo i naveli najveće znanstvenike koji su doprinijeli unapređenju ove grane umjetne inteligencije. Sadržaj idućeg poglavlja temeljen je na izvorima [8, 10, 18, 5].
1950. – Alan Turing kreira „*Turingov test*“ kako bi utvrdio ima li računalo stvarnu inteligenciju. Da bi prošao test, računalo mora biti sposobno prevariti čovjeka i uvjeriti ga kako je računalo čovjek.
1952. – Arthur Samuel napisao je prvi program računalnog učenja. Program je bio igra „dame“, a IBM-ovo računalo se, proučavajući koji su potezi rezultirali dobitnim strategijama i uvrštavajući ih u svoj program, sa svakom odigranom partijom poboljšavalo u igranju.
1957. – Frank Rosenblatt dizajnirao je prvu neuronsku mrežu za računala koja je simulirala misaone procese ljudskog mozga.
1967. – Napisan je algoritam „najbližeg susjeda“ koji je omogućio računalima početak korištenja osnovnog prepoznavanja uzoraka.
1979. – Studenti na sveučilištu Stanford izumili su „*Stanford Cart*“, robota koji se samostalno kreće izbjegavajući prepreke.
1981. – Gerald Dejong uvodi koncept učenja zasnovanog na objašnjavanju („*Explanation Based Learning*“) u kojem računalo analizira podatke sa treninga i stvara opće pravilo koje slijedi odbacivanjem nebitnih podataka.
1985. – Terry Sejnowski stvara NetTalk koji uči izgovarati riječi na isti način kao bebe.
1990-te – Rad na strojnom učenju prelazi s pristupa temeljenog na znanju na pristup temeljen na podacima. Znanstvenici počinju stvarati programe za računala kako bi analizirala velike količine podataka i izvodila zaključke, to jest „učila“ iz rezultata.
1997. – IBM-ov Deep Blue pobjeđuje u šahu svjetskog prvaka Garryija Kasparova.

2006. Geoffrey Hinton uvodi pojam „duboko učenje“ kako bi objasnio nove algoritme koji računalima omogućavaju uvidjeti i razlikovati predmete i tekst u slikama i videima.
2010. – Microsoft Kinect može pratiti 20 ljudskih svojstava brzinom od 30 puta po sekundi, omogućujući ljudima da komuniciraju s računalima pomoću pokreta i gestikulacija.
2011. – razvijen je Google Brain, njegova duboka neuronska mreža može naučiti otkrivati i kategorizirati objekte na jednak način kao i mačka
2012. – Googleov X laboratorij razvija algoritam koji može samostalno pregledavati videozapise na YouTubeu i prepoznati videozapise koji sadrže mačku
2014. – Facebook razvija DeepFace, softverski algoritam koji prepoznaje i provjerava pojedince na fotografijama na isti način kao i ljudi
2015. – Preko tri tisuće istraživača umjetne inteligencije i robotike, koje su podržali i Stephen Hawking, Elon Musk i Steve Wozniak, potpisuju otvoreno pismo upozoravajući na opasnost od autonomnog oružja koje bira i uključuje ciljeve bez ljudske intervencije
2016. – Googleov algoritam umjetne inteligencije pobijedio je profesionalnog igrača u kineskoj igri Go, koja se smatra najsloženijom svjetskom stolnom igrom. AlphaGo algoritam uspio je pobijediti u svih pet igara koje su igrali.

Kao što se vidi iz pregleda, disciplina strojnog učenja je relativno nova. Također je vidljivo kako su se njome početno bavili pojedinci, dok se razvojem tehnologije u istraživanje strojnog učenja uključuju svjetski poznate kompanije, što je potvrda koliko je strojno učenje važno te koliki potencijal leži u razvoju umjetne inteligencije.

2.2 Definicija i podjela strojnog učenja

Nakon što je prikazan pregled kroz povijest, logično je zapitati se što je točno strojno učenje. Mitchell je definirao strojno učenje: Kažemo da računalni program uči iz iskustva E u odnosu na neki skup zadataka T i s obzirom na mjeru uspješnosti izvođenja P, ako se povećava uspješnost obavljanja zadatak T, kroz iskustvo E, mjerena mjerom uspješnosti P. [10]

Na primjer, zamislimo da računalni program uči samostalno voziti. Kako bi ispravno identificirali problem učenja, potrebno je odrediti sljedeće značajke: zadatak, mjeru uspješnosti koja se treba unaprijediti i izvor iskustva. U nastavku slijedi opis za svaku značajku:

- Zadatak T: vožnja javnom prometnicom uz pomoć vizualnih senzora
- Uspješnost izvođenja P: prosječna prijeđena udaljenost prije pogreške u vožnji
- Iskustvo E: niz slika i komandi snimljenih promatranjem vozača (čovjeka)

Za bolje razumijevanje, izdvojena je i specifičnija definicija strojnog učenja: „Strojno učenje jest programiranje računala na način da optimiziraju neki kriterij uspješnosti temeljem podatkovnih primjera ili prethodnog iskustva.“ [2] U današnjem svijetu podataka ima u izobilju (internet, knjige, eksperimentalni podatci, itd.). S druge strane, znanja nema puno i ono je skupo, stoga je cilj izgradnja modela koji objašnjavaju podatke i omogućavaju zaključivanje/predviđanje. Upravo to čine algoritmi strojnog učenja: grade matematički model baziran na podatcima iz uzorka, koje još nazivamo i podacima za trening, kako bi napravio predviđanje ili donio odluku bez da je eksplicitno programiran za to. Podaci se dijele na skup za učenje i skup za testiranje u omjeru 70/30 ili 80/20. Iako ćemo u radu detaljno proći kroz princip rada i algoritam, u nastavku slijedi jednostavni prikaz kako se konstruira model strojnog učenja:

1. Uzimamo neki skup podataka za koji znamo odgovor
2. Treniramo algoritam na toj bazi podataka (nazivamo ju skup za učenje)
3. Uzimamo bazu podataka za koju želimo odgovor (nazivamo ju skup za testiranje)
4. Prenosimo te podatke kroz svoj obučeni algoritam i dobivamo rezultat

Vrste strojnog učenja

1. Nadzirano učenje
   Nadzirani algoritmi učenja grade matematičke modele skupa podataka koji sadrži ulazne i željene izlazne podatke. Dakle ulazni podatci su oblika (x, y), pri čemu je $x = (x_1, \ldots, x_n)$ vektor značajki, a y željena vrijednost. Cilj učenja je pronaći funkciju za koju vrijedi $f(x) = y$. To je tip učenja pod koji spada naš algoritam.
   Nadzirano učenje dijelimo na dva modela: klasifikaciju i regresiju.
2. Nenadzirano učenje
   Za razliku od nadziranog učenja, ovdje imamo samo ulazne podatke oblika $x = (x_1, \ldots, x_n)$ te je cilj pronaći pravilnosti među njima. Glavne metode ovakvog tipa učenja su grupiranje (eng. *clustering*), procjena gustoće (eng. *density estimation*) i smanjenje dimenzionalnosti (eng. *dimensionality reduction*).
3. Podržavano/ojačano učenje
   Radi se o učenju najbolje strategije na temelju pokušaja kako bi se maksimizirala kumulativna nagrada.

2.3 Primjena strojnog učenja

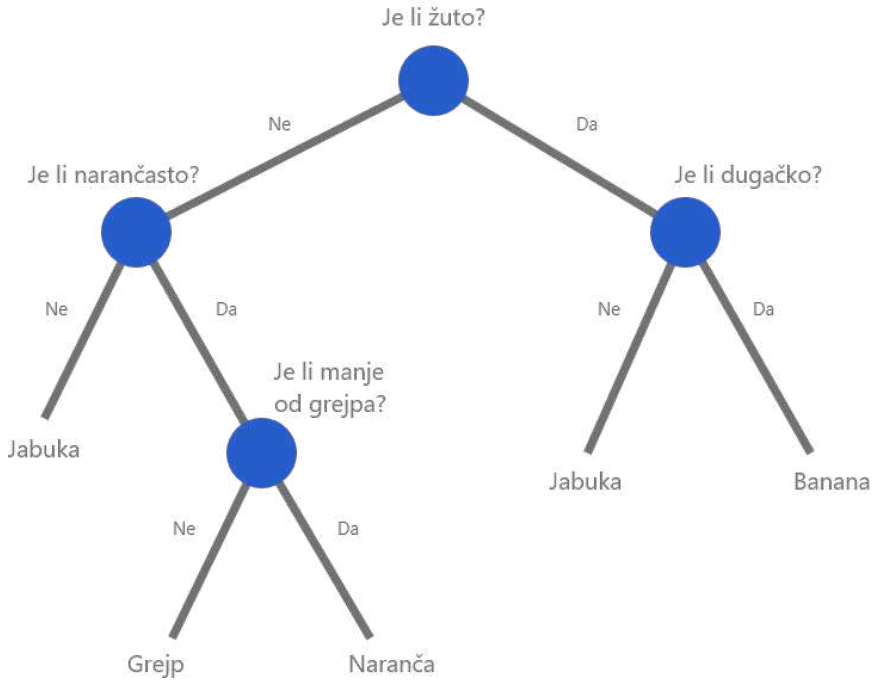
Ljudi vjerojatno to ne primjećuju, ali većina nas je svakodnevno u doticaju sa strojnim učenjem. Ovo je nekoliko popularnih primjera aplikacija i uređaja koje koriste algoritme strojnog učenja: Google (kako bi poboljšao preciznost tražilice), Facebook (kako bi prikazivao objave koje su bliže našim interesima i prethodnom ponašanju na društvenim mrežama), samovozeći automobili (pomoću algoritma prate obližnje objekte i koriste te informacije kako bi poboljšali sposobnosti).[5]
Strojno učenje se također primjenjuje u medicini. Pomoću njega se predviđa životni vijek, organiziraju pacijentovi podatci i dijagnosticiraju bolesti (npr. prikupljeni podatci krvnih testova, rentgentskih slika i sličnog mogu dati sigurniju i bržu dijagnozu, iako takva dijagnoza nije stopostotno sigurna).
Polje koje ima doticaj sa strojnim učenjem, a na koje ćemo se mi koncentrirati, je ekonomija. Algoritam se primjenjuje u bankarstvu za predviđanje kreditne sposobnosti i sprječavanje prijevare korisnika, online trgovini za predviđanje vjerojatnosti prodaje proizvoda te u financijama za predviđanje kretanja cijena dionica.
Prethodno navedene stvari se čine pozitivnima, no potrebno je napomenuti da takvi oblici unapređenja zanimanja imaju i negativne efekte. Sasvim je logično zapitati se hoće li ljudi zbog strojnog učenja gubiti poslove. Iako to ovisi o više čimbenika, s obzirom na široki raspon poslova koje bi algoritam mogao obavljati, jasno je kako bi računala u budućnosti mogla zamijeniti vozače, bankare te neke doktore.
Mnogi su zabrinuti zbog etičnosti strojnog učenja i njegove mogućnosti narušavanja privatnosti. Na primjer, nije neizvedivo kreirati program koji bi prikupljao naše poruke te se u razgovoru s drugom osobom predstavljao kao mi. Možemo zaključiti da je algoritam strojnog učenja jako moćan te bi trebali promisliti o njegovim pozitivnim i negativnim posljedicama prije implementiranja u razne sektore.

Poglavlje 3

Stablo odluke

3.1 Definicija i svojstva stabla odluke

Stablo odluke je vrsta nadziranog strojnog učenja u kojoj se podatci neprestano dijele prema određenim parametrima. Osim strojnog učenja, koristi se i u statistici te rudarenju podataka. Sadržaj idućeg poglavlja temeljen je na izvorima [18, 5].
Stablo se sastoji od čvorova, grana i listova. Svaki čvor predstavlja jednu značajku rješavanog problema. Grane koje idu iz čvora predstavljaju svaku moguću vrijednost koju taj čvor može poprimiti i u ovisnosti o vrijednosti tog atributa za određeni ispitni slučaj, putujemo kroz stablo određenom granom. Na kraju se nalaze listovi koji ujedno predstavljaju i razrede u koje možemo klasificirati određene ulazne podatke. Radi boljeg razumijevanja, slijedi jednostavni prikaz stabla odluke.
Na slici 3.1 čvorove čine: žuto, narančasto, dugačko, manje od grejpa. Grane čine odgovori da ili ne, a konačni listovi čine razrede u koje se ulazni podaci klasificiraju. Npr. u slučaju da voće nije žuto, da je narančasto i da je manje od grejpa izvela bi se grana koja označava razred naranči. Važno je naglasiti da čvorovi nisu morali nužno slijediti navedenim redoslijedom. Stabla odluke su jedna od najpraktičnijih metoda nadziranog učenja. Koriste se za klasifikaciju i regresiju. Modeli stabala u kojima se za ciljanu varijablu uzima diskretni skup nazivaju se klasifikacijska stabla. Stabla odluke u kojima se za ciljanu varijablu uzimaju kontinuirane vrijednosti (obično stvarni brojevi) nazivaju se regresijska stabla. Naravno da u stvarnom životu podaci neće biti tako jasno raspoređeni kao u našem primjeru, ali sistem stabla odluke ostaje isti. Na svakom čvoru će se pitati: Koje značajke će omogućiti takvu podjelu da rezultat budu grupe koje su različite što je više moguće i članovi podgrupa su što je više moguće međusobno slični?



Slika 3.1: Stablo odluke

3.2 Izgradnja stabla odluke

Postoji više algoritama izgradnje stabla odluke, a najvažniji su:

1. ID3
2. C4.5
3. CART

Koncentrirat ćemo se na metodu CART jer se ona koristi u radu. CART je oznaka za *„Classification and Regression Trees“*, što u prijevodu znači *„Klasifikacijska i regresijska stabla“*. Prvi put je upotrijebljena 1984. i konstruira jedino stabla s binarnim dijeljenjem, dok ID3 i C4.5 mogu konstruirati višestruka dijeljenja. Na primjer, zamislimo da stablo ispituje prihode. Kod algoritma CART modus operandi bio bi: ako je prihod $< 35k$ onda X, inače Y. Kod algoritama ID3 i C4.5 bili bi u mogućnosti detaljnije podijeliti čvor: ako je prihod $< 35k$ onda X, ako je prihod $> 35k$ i $< 50k$ onda Y, inače Z.
Kao i svaki algoritam, tako i CART, ima svoje dobre i loše strane. Pozitivno je što može raditi s kontinuiranim i kategorijalnim varijablama. To znači da se može koristiti i za regresiju i za klasifikaciju. Također je pozitivno što može raditi s nepotpunim podacima. No

ipak, može se dijeliti samo binarno, što je već prethodno navedeno kao jedna od karakteristika koja ga razlikuje od druga dva algoritma. Povrh toga, osjetljiv je na nestabilnost. Do nje dolazi kada listovi nisu 100% „čisti“.
Prethodno smo vidjeli vizualni prikaz stabla odluke, no promatrajući ga nameće se jedno veoma bitno pitanje. U prošlom odlomku navedeno je da čvorovi, to jest atributi, ne moraju nužno ići određenim redoslijedom. Kako onda stablo zna koji atribut ispitati u danom trenutku?
Cilj stabla je postići što veću razinu „čistoće“ (eng. *purity*). Čistoća je mjera kojom se stablo koristi kako bi odredio redoslijed dijeljenja čvorova. Također, čistoća je mjera stabilnosti. Vratimo se malo na naš primjer s voćem. Kada bi se u završnom listu, pod koji pripadaju naranče, pojavilo devet naranči i jedna jabuka, tada taj list ne bi bio 100% čist, čime se povećava nestabilnost stabla.
Dakle, atribut za podjelu se ne bira nasumično, već algoritam upotrebljava specifične alate kako bi odabrao atribut s najvećom čistoćom. Mjera čistoće se računa pomoću dobitka informacije (eng. *Information Gain*).

Definicija 3.2.1. *Dobitak informacije dobiven odabirom baš tog atributa A_i za grananje ili particiju podataka je smanjenje entropije. Do njega dolazimo putem sljedeće formule:*

$$dobitak(D, A_i) = entropija(D) - entropija_{A_i}D[15] \tag{3.1}$$

Dobitak informacija možemo još opisati kao mjeru koliko informacija o razredu daje značajka. Stablo odluke uvijek pokušava maksimizirati dobit informacija te će se zato atribut sa najvećim dobitkom informacija dijeliti prvi (to je onaj koji ima najmanju entropiju). Dijeljenjem čvorova dobivamo dodatne informacije o uzorku i time se svakim dijeljenjem dodatno smanjuje entropija.

3.3 Entropija

Lako se primijeti da je za izračun dobitka informacija potreban iznos entropije određenog atributa. Entropija je mjera nečistoće u skupu podataka. Računamo je, za slučajnu varijablu V, s vrijednostima v_k pomoću sume

$$H(V) = -\sum_k P(v_k) \cdot \log_2 P(v_k), \sum_k P(v_k) = 1 \tag{3.2}$$

gdje je $P(v_k)$ vjerojatnost ishoda v_k.
Pokažimo kako dobitak informacija funkcionira na novom primjeru s voćem. Imamo četiri vrste voća koje je potrebno klasificirati u četiri kategorije. Kategorije su banana, naranča, jabuka i limun. Pretpostavimo da svaki uzorak ima tri atributa: boja, tekstura i oblik te da imamo 200 uzoraka za treniranje, pri čemu uzorak predstavlja jedno od navedenog voća. Zamislimo da imamo četrdeset banana, pedeset naranči, šezdeset jabuka i pedeset limuna. Za početak možemo izračunati entropiju početnog čvora pomoću 3.2:

$$E(\text{roditelj}) = -\tfrac{1}{5} \cdot \log_2 \tfrac{1}{5} - \tfrac{1}{4} \cdot \log_2 \tfrac{1}{4} - \tfrac{3}{10} \cdot \log_2 \tfrac{3}{10} - \tfrac{1}{4} \cdot \log_2 \tfrac{1}{4} = 1.99$$

Sada ćemo, s obzirom koji atribut izaberemo, imati različite entropije u sljedećem čvoru. Kako bi stablo odabralo najprimjereniji čvor računa entropiju svakog od atributa te se odabire onaj s najvećom dobiti informacija. Čvor koji ispituje boju podijelit će uzorke na stodeset uzoraka u jednom čvoru (voće koje nije žuto, označeno s NY) i devedeset uzoraka u drugom čvoru (voće koje je žuto, označeno s Y). Izračunajmo entropiju ta dva čvora.

$$E(Y) = -\frac{50}{200} \cdot \log_2 \frac{50}{200} - \frac{40}{200} \cdot \log_2 \frac{40}{200} = 0.96$$

$$E(NY) = -\frac{60}{200} \cdot \log_2 \frac{60}{200} - \frac{50}{200} \cdot \log_2 \frac{50}{200} = 1.02$$

Ukupna entropija novih čvorova je:

$$E(\text{boja}) = -\frac{90}{200} \cdot 0.96 + \frac{110}{200} \cdot 1.02 = 0.99$$

Provedimo isti izračun za preostala dva atributa, teksturu i oblik.

Dijeljenje po teksutri razdijelit će uzorke u dva čvora sa sto uzoraka u svakom čvoru (hrapavo koje je označeno s H i nije hrapavo sa NH).

$$E(H) = -\frac{50}{200} \cdot \log_2 \frac{50}{200} - \frac{50}{200} \cdot \log_2 \frac{50}{200} = 1$$

$$E(NH) = -\frac{50}{200} \cdot \log_2 \frac{50}{200} - \frac{50}{200} \cdot \log_2 \frac{50}{200} = 1$$

Ukupna entropija novih čvorova je:

$$E(\text{hrapavo}) = \frac{100}{200} \cdot 1 + \frac{100}{200} \cdot 1 = 1$$

Preostaje izračunati entropiju za oblik. Ako se uzorci dijele po tom atributu, podijelit će se u dva čvora sa četrdeset (duguljasti oblik označen s D) i sto šezdeset uzoraka (ne duguljasti oblik označen s ND).

$$E(D) = -\frac{40}{200} \cdot \log_2 \frac{40}{200} = 0.46$$

$$E(ND) = -\frac{50}{200} \cdot \log_2 \frac{50}{200} - \frac{50}{200} \cdot \log_2 \frac{50}{200} - \frac{60}{200} \cdot \log_2 \frac{60}{200} = 1.52$$

Ukupna entropija iznosi:

$$E(\text{oblik}) = \frac{40}{200} \cdot 0.46 + \frac{160}{200} \cdot 1.52 = 1.31$$

Zatim izračunajmo dobitak informacije.

$$I(\text{roditelj,boja}) = E(\text{roditelj}) - E(\text{boja}) = 1.99 - 0.99 = 1$$

$$I(\text{roditelj,tekstura}) = E(\text{roditelj}) - E(\text{tekstura}) = 1.99 - 1 = 0.99$$

$$I(\text{roditelj,oblik}) = E(\text{roditelj}) - E(\text{oblik}) = 1.99 - 1.31 = 0.68$$

Iz računa je vidljivo kako podjela uzoraka po atributu boje daje najveću dobit informacija zbog čega je taj atribut prvi izabran za dijeljenje.
Istim načinom dijelimo novonastale čvorove dok entropija ne iznosi 0. U odlomku *Definicija i svojstva stabla odluke* definirali smo listove kao završne dijelove stabla i upravo oni čine klasifikacijske razrede (banana, limun, naranča, jabuka). U našem primjeru u listovima entropija iznosi 0, no to ne mora uvijek biti slučaj.

Drugi kriterij kojim stablo može dijeliti čvorove je pomoću Gini indeksa. Gini indeks čvora je vjerojatnost da će nasumice odabran uzorak u čvoru biti netočno klasificiran. Na primjer, ako je Gini jednak 0.44, to znači da postoji 44% šanse za krivom klasifikacijom nasumice odabranog uzorka iz tog čvora). Gini indeks se, kao i entropija, računa za svaki čvor. Formula za Gini glasi:

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2, \tag{3.3}$$

gdje je n trenutni čvor, p_i je vjerojatnost klase i u čvoru n, a J je broj klasa u modelu. Također, kao za entropiju, što je indeks manji to je bolje, to jest čvor je sigurniji. Ako u listu iznosi 0, tada je svaki čvor kompletno čist i ne postoji šansa da nasumice odabran uzorak iz čvora bude krivo klasificiran. To se može činiti pozitivnim, ali potencijalno može doći do prenaučenosti (eng. *overfitting*).

Poglavlje 4

Slučajna šuma

4.1 Algoritam slučajne šume

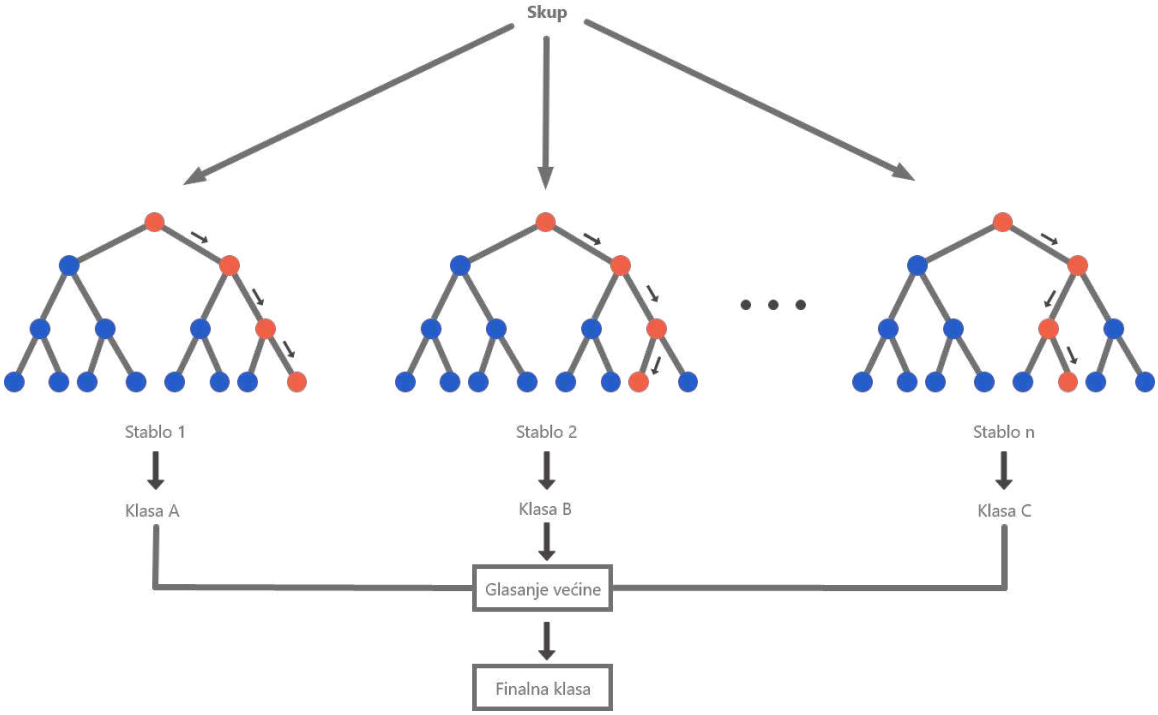
Algoritam slučajnih šuma je modifikacija *bagging* metode (više u poglavlju 4.2) koji gradi veliki broj nekoreliranih stabala. Pri klasifikaciji stabla predstavljaju glasove te algoritam očitava najmnogobrojniji glas (slika 4.1), dok pri regresiji algoritam pronalazi prosječnu vrijednost. Na mnogim problemima su izvedbe slučajnih šuma jako slične *boostingu* (metoda koja se temelji na permutacijama osnovnih modela) jednostavne su za treniranje i podešavanje. Kao posljedica, slučajne šume su jako popularan algoritam. Teoretski dio poglavlja temeljen je na sadržaju [6].

Prosječna varijanca B slučajnih varijabli, gdje je varijanca pojedinačne varijable σ^2, iznosi $\frac{1}{B}\sigma^2$. Ako su varijable jednako distribuirane sa međusobno pozitivnom korelacijom ρ tada je prosječna varijanca

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2. \tag{4.1}$$

Cilj algoritma slučajnih šuma je reducirati varijancu tako što će smanjiti korelaciju između stabala. Taj cilj se postiže kroz izgradnju stabala pomoću nasumično odabranih uzoraka (više u poglavlju 4.4). Za svaki uzorak se gradi posebno stablo prema sljedećem algoritmu:

1. Za $b = 1$ do B, gdje je B broj stabala
    a) Izvadi *bootstrap* uzorak Z^* veličine N iz skupa za učenje
    b) Izgradi stablo slučajne šume T_b trenirano na uzorku bootstrap, rekurzivno ponavljajući sljedeće korake za svaki čvor u stablu
        i. Izaberi m slučajnih varijabli od p mogućih.
        ii. Pronađi najbolju varijablu i vrijednost za podjelu među odabranim m.



Slika 4.1: Slučajna šuma - klasifikacija

iii. Podijeli čvor u lijevi i desni.

2. Spremi skup stabala $\{T_b\}_1{}^B$.

3. Predviđanje za novu točku x:

   a) Za regresiju: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

   b) Za klasifikaciju: $\hat{C}_{rf}^B(x) =$ najviše glasova$\{\hat{C}_b(x)\}_1{}^B$, gdje je $\hat{C}_b(x)$ predikcija pojedinog stabla.

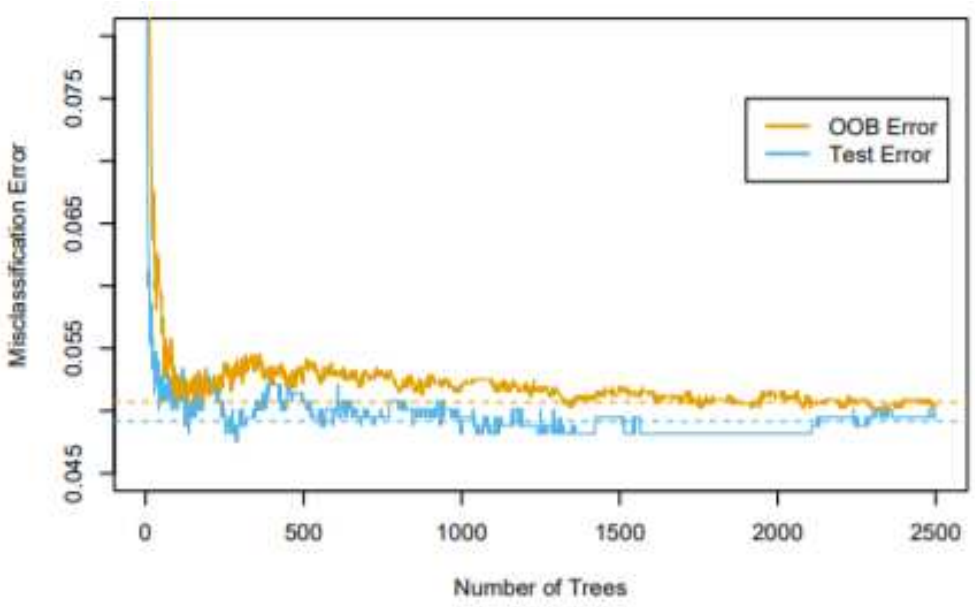
Nakon što se razvije B stabala $\{T(x;\Theta_b)\}_1^B$, predviđanje regresijske slučajne šume je:

$$\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T(x;\Theta_b), \tag{4.2}$$

gdje je Θ_b b-to stablo slučajne šume. Smanjenjem broja m smanjuje se korelacija između parova stabala i time se smanjuje prosječna varijanca. [6]

4.2 Out Of Bag

Važna stavka u algoritmu slučajnih šuma je ”out of bag” skup podataka. On predstavlja otprilike 33% podataka koji se zanemaruju pri uzorkovanju. OOB procjena greške je gotovo jednaka kao unakrsna validacija stabala, stoga je zadovoljavajuća i za procjenu greške slučajne šume. S obzirom da se algoritam provodi u jednoj sekvenci, unakrsna validacija se provodi istodobno kao i izgradnja stabala. U momentu kada se OOB greška stabilizira, trening podataka se prekida.



Slika 4.2: Usporedba greške OOB skupa i greške skupa za testiranje

Na slici 4.2 vidimo da se OOB greška stabilazirala oko 200-tog stabla.

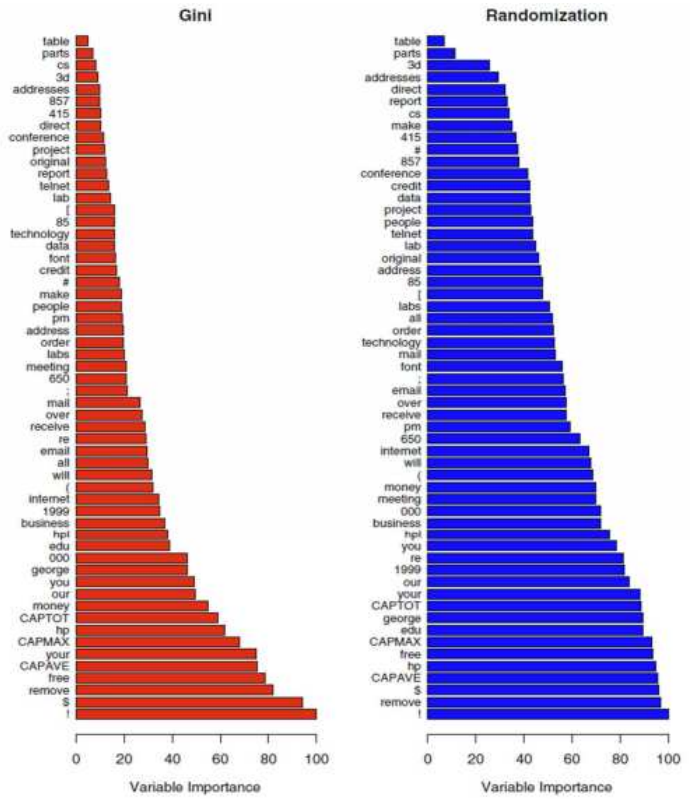
Nasumični uzorci podataka za treniranje pri gradnji stabla

Prethodno je napisano kako se 33% podataka ostavlja sa strane. Valjano je zapitati se kako algoritam odabire te podatke. Pri treniranju podataka svako stablo uči iz nasumičnog uzorka. Stabla odluke su jako osjetljiva na skupu podataka za treniranje pa male promjene na njemu mogu rezultirati mnogo drugačijim strukturama stabala. Slučajne šume iskorištavaju to tako što dopuštaju svakom individualnom stablu da nasumice uzme uzorak iz podataka zamjenom (eng. *replacement*), što rezultira različitim stablima.
Dakle kada imamo uzorak N, svako stablo uzima skup za učenje veličine N. Npr. imamo

skup (1, 2, 3, 4, 5, 6) te tada nasumično stablo može uzeti set za učenje (1, 2, 2, 3, 6, 6). Primijetimo da obje liste imaju duljinu 6 (šest članova). Ideja je da treniranje svakog stabla različitim uzorcima, iako će svako stablo imati veliku varijancu s obzirom na određeni skup trening podataka, rezultira manjom varijancom cijele šume. Na testu, predviđanja se rade prema prosjeku predviđanja svakog stablo pojedinačno. Ovaj proces treniranja svakog individualnog „učenika“ na različitom podskupu podataka i onda uzimanje prosjeka predviđanja se naziva *bagging*.

4.3 Važnost varijance

U poglavlju 3.3 opisano je kako stabla odluke računaju važnost pojedinih atributa pomoću kriterija entropije ili Gini indeksa. Slučajne šume, osim navedenih kriterija, obavljaju procjenu važnosti pojedinih atributa pomoću OOB skupa. Kada b-to stablo raste, OOB skup podataka se prenosi kroz stablo i bilježi se točnost predviđanja. Zatim vrijednosti j-te varijable nasumično permutiraju i ponovno se provodi izračun preciznosti stabla. Smanjenje preciznosti, kao rezultat permutacije, se pamti te se računa prosječna vrijednost za sva stabla. Atribut ima veću važnost ako je smanjenje preciznosti čim veće. Na slici 4.3 važnost je izražena u postotcima. Lijevi graf prikazuje važnost varijabli izračunatih pomoću Gini indeksa, a desni graf prikazuje važnost varijabli izračunatih pomoću OOB randomizacije. Iako obje metode rezultiraju sličnim vrijednostima, važnosti na desnoj slici su više uniformne. Također, ako bismo jedan atribut uklonili u potpunosti, pomoću metode OOB ne možemo donijeti zaključak kako bi se ostali atrbituti odnosili jer bi tada stablo bilo izgrađeno na posve različit način.

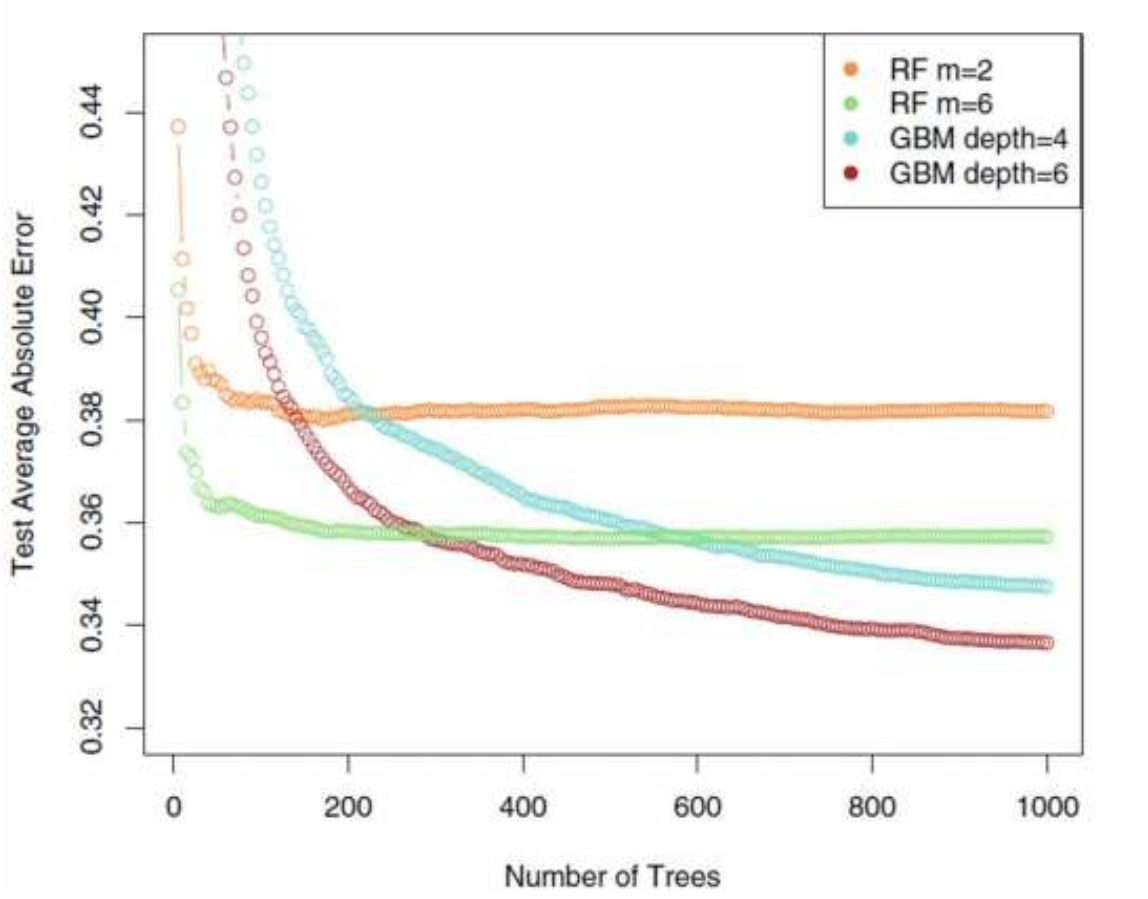


Slika 4.3: Procjena važnosti atributa

4.4 Prenaučenost

Pri radu algoritma stabla odluke, a samim time i slučajne šume, mogu nastupiti dva problema pod nazivima prenaučenost (*eng. overfitting*) i podnaučenost (*eng. underfitting*). Prenaučenost se pojavljuje češće stoga ćemo se u ovom poglavlju više posvetiti njoj.

Prenaučenost (u literaturi se pojavljuje i naziv prekomjerna specijalizacija) se javlja kada funkcija savršeno opisuje podatke, ali upravo zbog toga ne predviđa dobro za novi skup podataka. Do nje dolazi kada imamo složenu funkciju koja uzorkuje mnogo beskorisnih putanja specifičnih baš za tu krivulju. Samim time, na novoj (različitoj) funkciji će model imati nisku vrijednost pristranosti i neće točno klasificirati nove podatke [2]. Osnovna ideja *bagginga* je bilježenje mnogo šumovitih, ali nepristranih podataka kako bi model bio što precizniji prilikom obrade novih podataka. Stabla odluke su podobna za *bagging* jer prepoznaju složene strukture između atributa i klasa. Budući da je svako stablo jednako distruibuirano, očekivanje cjelokupnog ansambla je jednako kao i očekivanje svakog od stabala pojedinačno. To znači da je pristranost slučajne šume ista kao za individualna stabla, stoga je jedino moguće poboljšanje modela kroz smanjenje varijance.
Problem prenaučenosti se rješava slučajnim odabirom atributa pri izgradnji stabala.
Naime, kada imamo čvor, razmatramo svaku moguću značajku i biramo onu koja proizvodi najviši stupanj separacije između uzoraka (stavki) u lijevom i desnom čvoru. Ovaj postupak se radi kao algoritam klasteriranja (*eng. clustering*). Mjera koja se uzima za razdvajanje može biti Gini indeks ili dobitak informacija. No, kada bi sva stabla uvijek promatrala iste značajke bila bi jako slična. Stablo odluke ne zanima podjela između dvije grupe u postotcima (svejedno je li $50\% - 50\%$ ili $90\% - 10\%$). Jedino ga zanima da su slični (ovisno o značajkama) članovi zajedno i koliko je različit prosječan član jedne grupe s prosječnim članom druge grupe. Dakle, algoritam ne razmatra rezultate dijeljenja, nego pokušava maksimizirati sličnosti unutar grupe i minimizirati sličnosti između grupa. Suprotno, u slučajnim šumama svako stablo bira samo iz broja m nasumičnih podskupova značajki, gdje je $m \leq p$ (p je ukupan broj značajki). U pravilu se za m pri regresiji uzima $\lfloor p/3 \rfloor$ i minimalna veličina čvora je 5. Za klasifikaciju je za m najčešća vrijednost $\lfloor \sqrt{p} \rfloor$ i minimalna veličina čvora je 1. Uzimanjem manjeg broja značajki je način na koji se dodaje nasumičnost rezultatu i izbjegava prenaučenost. Broj zadržanih značajki za svako stablo podesivi je parametar i najbolja vrijednost može varirati ovisno o bazi podataka i problemu (slika 4.4). Na primjer, slučajne šume se mogu trenirati i da se uzimaju u obzir sve značajke kod svakog čvora.



Slika 4.4: Usporedba različitih parametara m naspram preciznosti algoritma

Za kraj poglavlja te kao sažetak napisanog o algoritmu slučajnih šuma sumirajmo cjelokupan princip rada algoritma:

1. Uzmi primjere i načini $\boldsymbol{T}$ nezavisnih "bootstrap" sampliranja (36,8% neizvučenih primjera služi za estimaciju pogreške). Svako nezavisno sampliranje (63,2% primjera) je osnovica za učenje jednog stabla odlučivanja.

2. Izgradi maksimalno duboka stabla (bez podrezivanja). U izgradnji stabala za svaki čvor slučajno odaberi mali podskup značajki. Unutar toga podskupa odredi najbolju značajku uobičajenom metodom (entropija ili Gini indeks).

3. Podskup stabala testiraj s njihovim zajedničkim neizvučenim primjerima ili svako stablo testiraj s njegovim neizvučenim primjerima. Usrednji pogrešku preko svih stabala.

4. Za nepoznati primjer X prikupi glasove svih stabala i odredi većinski razred klasifikacije.

Objašnjenja:

- Povećavati broj stabala ($\boldsymbol{T}$) dok se točnost klasifikacije ne ustali.
- Veličina podskupa slučajnih značajki za svaki čvor je kvadratni korijen iz cijelog skupa. Povećanje toga slučajnog podskupa povećava točnost klasifikacije svakog pojedinog stabla ali i međusobnu korelaciju između stabala što smanjuje točnost klasifikacije cijele šume. [1]

Poglavlje 5

Primjena

5.1 Opis problema

Mnogi ljudi se, zbog nedovoljne ili nepostojeće kreditne povijesti, bore da dobiju zajmove. Nažalost, tu skupinu ljudi često iskoriste nepovjerljivi zajmodavci. Stoga je prirodna tendencija ljudima koji nisu imali doticaja s bankama pružiti pozitivno i sigurno iskustvo sa zaduživanjem. Kako bi se približili željenom cilju, za procjenu kreditne sposobnosti klijenta koriste se razni alternativni podaci. S obzirom da klijent nema povijest zaduživanja, u procjenu se uključuju telekomunikacijski podaci i podaci o bankovnim transakcijama. Za izradu predviđanja koriste se razne statističke metode i metode strojnog učenja te je jedna od njih i model slučajnih šuma. Pomoću njega se osigurava da klijenti sposobni za otplatu ne budu odbijeni.

5.2 Opis podataka

Opisna statistika

Postoji sedam različitih izvora podataka preuzetih s [4]:

- **Prijave-trening/prijave-test**
  Glavni podaci za trening sadrže informacije o svim prijavama za kredit. Svaki zajam ima svoj red i prepoznaje se pomoću značajke ”SK-ID-CURR”. Trening podaci za prijave su označeni s brojevima 0 ili 1: broj 0 ima značenje da je kredit otplaćen, a 1 da nije otplaćen. U našem algoritmu ćemo koristiti isključivo podatke za trening.
- **Biro**
  Ovaj skup podataka sadrži informacije koje se odnose na prethodne kredite klijenta

pozajmljene u drugim financijskim institucijama. Svaki prijašnji kredit ima vlastiti red u *birou*, ali jedan zajam u podatcima *prijave* može imati više prethodnih kredita.

- **Saldo biroa**
  Sastoji se od mjesečnih podataka o prethodnim kreditima u birou. Svaki red predstavlja jedan mjesec kredita. Pojedinačni prethodni kredit može imati više redova, po jedan za svaki mjesec trajanja kredita.

- **Prethodna prijava**
  Skup podataka sadrži informacije o prijašnjim zahtjevima za kredite. Svaki trenutni zajam u podacima o prijavama može imati više prethodnih zajmova. Svaka prethodna aplikacija ima jedan red i može se prepoznati pomoću značajke ”SK-ID-PREV”.

- **POS-CASH-saldo**
  Sastoji se od mjesečnih podataka o prethodnom prodajnom mjestu ili gotovinskih zajmova klijenta. Svaki red predstavlja jedan mjesec gotovinskog zajma te jedan prethodni zajam može imati više redova.

- **Saldo kreditne kartice**
  Mjesečni podaci o prethodnim kreditnim karticama klijenta. Svaki red predstavlja saldo kreditne kartice u pojedinačnom mjesecu i pojedinačna kartica može imati više redova.

- **Plaćanje rate**
  U ovom skupu nalaze se podaci o povijesti plaćanja prethodnih zajmova. Za svaku izvršenu uplatu postoji jedan red. Također postoje redovi i za svaku propuštenu uplatu.

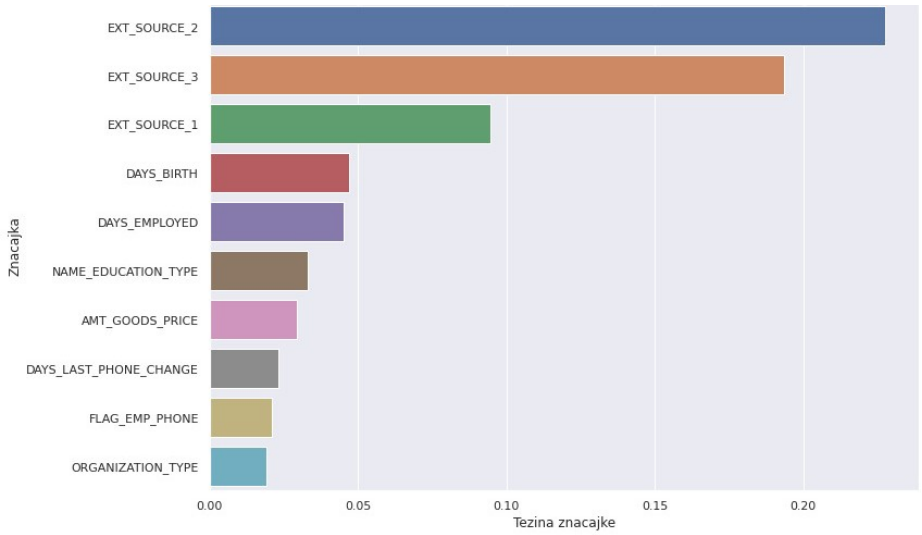
Konstrukcija složenih ulaznih podataka

Naš algoritam napravljen je u programu Python. Prije nego što opišemo algoritam, napravljen je kratak pregled podataka potkrijepljen grafovima i histogramima kako bi olakšali analizu i modeliranje. Podaci se sastoje od 307 511 redova i 122 stupca. Svaki red ima jedinstveni identifikacijski broj (”SK-ID-CURR”), a izlazna varijabla se nalazi u stupcu ”TARGET”. U stupcu ”TARGET” nula označava da je kredit otplaćen, a jedan suprotno, tj. da kredit nije otplaćen (slika 5.1).

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_
0	100002	1	Cash loans	M	N	Y	0	202500.0	
1	100003	0	Cash loans	F	N	N	0	270000.0	1
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	
3	100006	0	Cash loans	F	N	Y	0	135000.0	
4	100007	0	Cash loans	M	N	Y	0	121500.0	

Slika 5.1: Raspoređeni podaci

S obzirom da imamo mnogo značajki, korisno je odrediti koje značajke su najvažnije. Stoga smo odredili deset najznačajnih značajki. Na slici 5.2 vidimo da postoji nekoliko

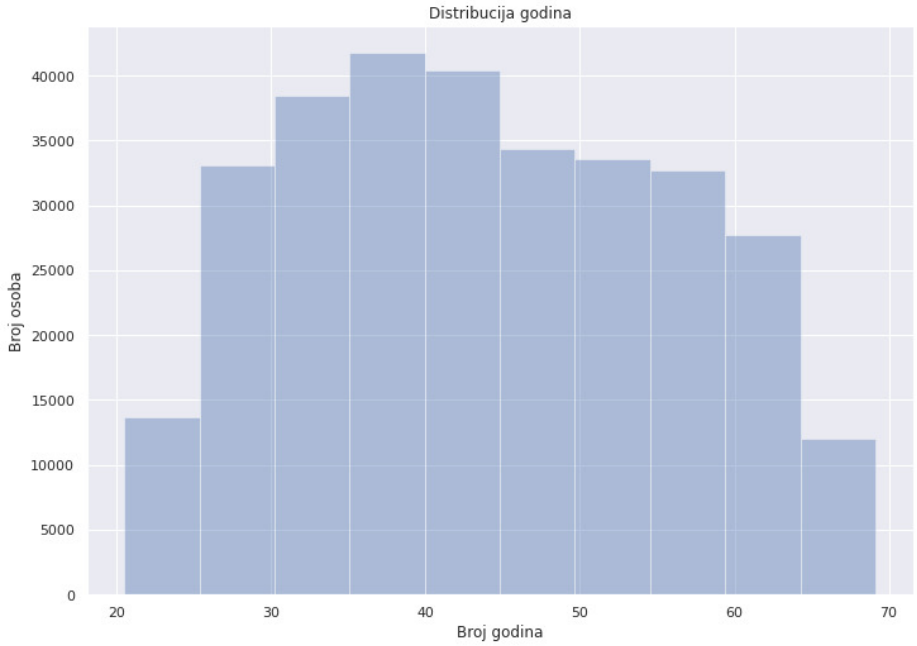


Slika 5.2: Deset najznačajnih značajki

značajki koje imaju veliku važnost za model, što sugerira da se od mnogih može odustati bez smanjenja performansi modela (možda bi se performanse i povećale). Važnost značajki nije najbolja metoda za tumačenje kompletnog modela, ali mogu nam pomoći da shvatimo koje faktore naš model uzima u obzir kada vrši predviđanje.

Jedna od važnijih značajki je distribucija godina. Sa slike 5.3 vidimo da najviše ljudi uzima zajam u četvrtom desetljeću života. Ovaj podatak je logičan jer se pretpostavlja da su se ljudi do tada ustabilili na poslu, imaju sigurna primanja i mnogo godina radnog vijeka ispred sebe. S toga ne čudi da se dvadesetogodišnjaci ne zadužuju u tolikoj mjeri jer većina njih u tim godinama studira ili tek kreće raditi. Također, ljudi u mirovini rijetko uzimaju

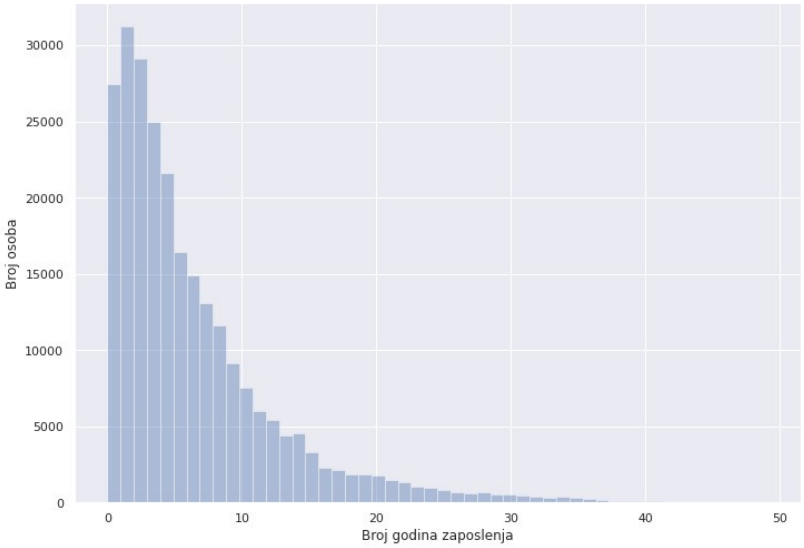
zajmove jer im je platežna moć manja nego za vrijeme radnog vijeka te su, s obzirom na godine, lišeni mogućnosti dugoročnih kredita.



Slika 5.3: Distribucija godina

Sljedeća značajka po važnosti je duljina zaposlenja dužnika. Iz slike 5.4 jasno je vidljivo kako je tendencija ljudi uzimati kredite u prvim godinama radnog vijeka, točnije u prvih pet godina. Broj zajmoprimaca je obrnuto proporcionalan duljini radnog vijeka zajmoprimca.

Još jedna značajka u nizu najznačajnijih je razina obrazovanja dužnika (slika 5.5). Možemo zaključiti kako se najčešće zadužuju građani sa sekundarnim obrazovanjem. Iz grafikona je također vidljivo kako akademski obrazovani građani rijetko uzimaju kredite.

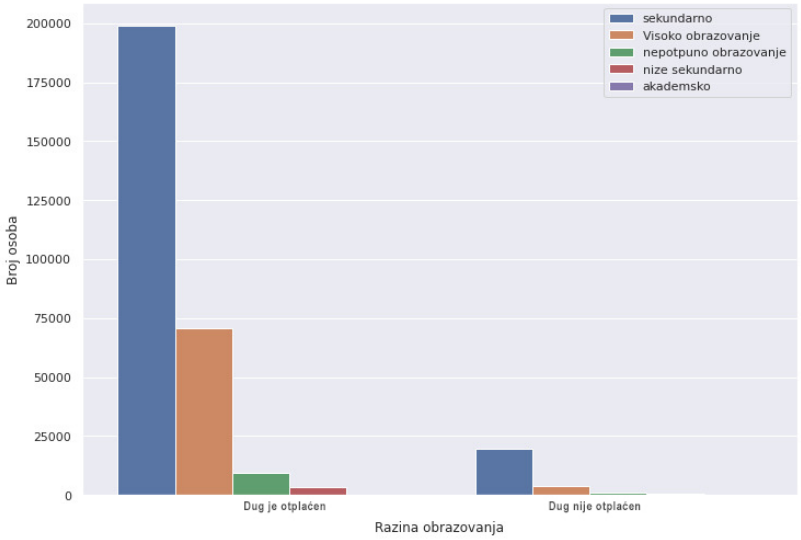


Slika 5.4: Duljina radnog staža dužnika

5.3 Pristup modeliranju i rezultati

Opis primijenjene metode

Za početak, ubacujemo sve Python pakete nužne za rad algoritma. Pandas je brza i fleksibilna biblioteka za učitavanje csv-a i manipulaciju podataka. Numpy je matematička biblioteka za rad s velikim, multidimenzionalnim vektorima i matricama. Sklearn je biblioteka za strojno učenje. Confusion matrix predstavlja tablični sažetak broja točnih i netočnih predviđanja. Roc-auc-score i accuracy-score predstavljaju izračun ROC-AUC vrijednosti te točnost algoritma. Njihovu implementaciju i svrhu ćemo detaljnije objasniti u sljedećem potpoglavlju. Naredba OrdinalEncoder služi za enkodiranje klasa u vektor. SimpleImputer koristimo kako bi nadopunili vrijednosti koje nedostaju. RandomForestClassifier je temeljni algoritam koji koristimo, a train-test-split je metoda pomoću koje razdvajamo skup podataka. Zatim učitavamo podatke i uzimamo *y* vrijednosti. Algoritam prvo stupac ”TARGET” pretvara u *y* vrijednosti. Uklanja identifikacijske brojeve s podataka (SK-ID-CURR) i dijeli podatke na setove za treniranje i testiranje. Zatim obrađuje podatke kako bi bili prikladni za treniranje; među podacima se nalazi dosta praznih polja u stupcima, odnosno nedostaje dio podataka. Njih moramo popuniti ovisno radi li se o kategorijskim ili brojčanim ćelijama. Ako podaci nedostaju u kategorijskim stupcima, prazne ćelije se ispunjavaju riječju ”Missing”. Zatim konvertiramo kategorijske značajke u brojeve. U suprotnom, nedostaju li podaci u brojčanim ćelijama, one se ispunjavaju najčešćim vrijednostima. Podaci se nakon transformacija ponovno kombiniraju zajedno u tablice.



Slika 5.5: Razina obrazovanja dužnika

Nakon pripreme podataka slijedi traženje najboljih parametara; potom se kreira mapa za sve rezultate te se inicijaliziraju varijable za najbolji AUC i najbolje parametre. Zatim se za svaki kriterij (entropija i Gini), broj procijenitelja (3, 6, 50, 100 i 200) i maksimalnu dubinu (3,6,10,50,100) kreira klasifikator slučajnih šuma s danim parametrima. Algoritam će se natrenirati na skupu za učenje i odrediti rezultat na skupu za testiranje. Za svaki primjer [p_1,p_2] iz skupa podataka računaju se vjerojatnosti p_1 i p_2, gdje je p_1 vjerojatnost da rezultat bude 0, a p_2 vjerojatnost da rezultat bude 1. Ovdje su nam mnogo bitniji rezultati za p_2, odnosno vjerojatnosti da primjer spada u klasu 1, jer su potrebni za ROC-AUC mjeru.
Nakon toga računaju se točnost algoritma i ROC-AUC rezultat (zbog kojeg smo računali vjerojatnosti). Ako je novi ROC-AUC rezultat veći od najboljeg, algoritam sprema novi najbolji AUC. Također sprema nove najbolje parametre.
Nakon pronalaska najboljih parametara slijedi učenje najboljeg klasifikatora. Poslije učenja određujemo točnost klasifikatora na skupu za testiranje te matricu konfuzije. Sada kada imamo sve parametre relevantne za rad algoritma, može se provesti trening najboljeg klasifikatora. Za kraj se ispisuju podaci o točnosti.

Točnost modela na testnim podacima

Provedbom opisanog algoritma dolazimo do nekoliko bitnih podataka. U ovom dijelu ćemo ih iznijeti te protumačiti njihovo značenje za cjelokupan postupak.
U prethodnom potpoglavlju na nekoliko mjesta su spomenuti izrazi ROC (*eng. Receiver Operating Characteristics*) i AUC (*eng. Area Under the Curve*). ROC je krivulja koja

prikazuje odziv (*eng. True Positive Rate*) kao funkciju *fall-outa* (*eng. False Positive Rate*). Model koji radi nasumičnu klasifikaciju nalazit će se na pravcu TPR = FPR te će bolji model biti iznad, a lošiji ispod tog pravca. Vrijednost AUC odgovara površini ispod ROC krivulje. Što je površina veća to je model bolji [17].
Povucimo paralelu s našim algoritmom. U njemu smo tražili parametre za koje je AUC rezultat najveći te smo njih usvajali kao najbolje. Zatim smo s tim, najpodobnijim parametrima, provodili daljnji postupak. To smo napravili upravo zbog toga što smo time dobili najbolji mogući model.
Konkretno, naš AUC rezultat iznosi $0,73449$. Raspon mogućih rezultata je od 0 do 1. Nažalost, ne postoji "magična" brojka za zadovoljavajući AUC rezultat, samo općenite smjernice. Općenito, koristimo pravilo palca:

- $0,5$ = na temelju ovog rezultata ne donosimo bitne zaključke
- $0,5 - 0,7$ = smatramo lošim rezultatom
- $0,7 - 0,8$ = dobar rezultat
- $0,8 - 0,9$ = odličan rezultat
- $> 0,9$ = izvrstan rezultat. [7]

Prema tome, naš AUC pripada skupini dobrih rezultata. Uz njega su usvojeni i odgovarajući parametri. Sukladno tome dobiveni su sljedeći rezultati:

- broj procjenitelja iznosi 200
- maksimalna dubina iznosi 100
- kriterij je entropija.

Uz tako postavljene parametre, algoritam postiže točnost od $0,91928$, odnosno 91%. Što taj rezultat točno znači u našoj situaciji?
Kada netko dođe u banku i želi podići kredit, njegove informacije se provuku kroz model kako bi se donijela odluka. Na temelju modela, kreditor ima 91% šanse točno predvidjeti sposobnost zaduženika da vrati dug. Ovaj rezultat, odnosno preciznost, je solidan. Naravno da banke teže što manjem riziku stoga "nemaju hrabrosti" osloniti se samo na algoritam. S povećanjem baze podataka sigurno bi se postigli bolji rezultati, s obzirom modeli strojnog učenja uče iz iskustva. Dakle, sa svakim dodatnim podatkom iz skupa za treniranje model će napredovati.

Poglavlje 6

Dodatak

6.1 Algoritam

```
import pandas as pd
import numpy as np

from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OrdinalEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

def preprocess_dataset(x_train, x_test):
    # Get category columns
    x_train_cat = x_train.select_dtypes(include='object')
    x_test_cat = x_test.select_dtypes(include='object')

    train_cat_columns = x_train_cat.columns
    test_cat_columns = x_test_cat.columns

    # fill the missing values with "Missing"
    categorical_imputer = SimpleImputer(strategy='constant',
    fill_value='Missing')
    x_train_cat_with_missing = categorical_imputer.fit_transform
    (x_train_cat)
```

```
x_test_cat_with_missing = categorical_imputer.fit_transform
(x_test_cat)

# Convert categorical features into numbers
# "Cash loans" becomes '0', Revolving loans becomes '1' etc...
ordinal_encoder = OrdinalEncoder()
x_train_cat_with_missing_values_encoded =
ordinal_encoder.fit_transform
(x_train_cat_with_missing)
x_test_cat_with_missing_values_encoded =
ordinal_encoder.transform
(x_test_cat_with_missing)

# Get number features
x_train_num = x_train.select_dtypes(include='number')
x_test_num = x_test.select_dtypes(include='number')

train_num_columns = x_train_num.columns
test_num_columns = x_test_num.columns

# fill the missing features with most frequent
numeric_imputer = SimpleImputer(strategy='most_frequent')
x_train_num_with_missing = numeric_imputer.fit_transform
(x_train_num)
x_test_num_with_missing = numeric_imputer.transform
(x_test_num)

x_train_cat_with_missing_values_encoded = pd.DataFrame
(x_train_cat_with_missing_values_encoded,
columns=train_cat_columns)
x_test_cat_with_missing_values_encoded = pd.DataFrame
(x_test_cat_with_missing_values_encoded,
 columns=test_cat_columns)

x_train_num_with_missing = pd.DataFrame(x_train_num_with_missing,
columns=train_num_columns)
x_test_num_with_missing = pd.DataFrame(x_test_num_with_missing,
columns=test_num_columns)
```

```
    # Combine all feature back together after transformations
    x_train_comb = pd.concat([x_train_cat_with_missing_
    values_encoded,
    x_train_num_with_missing], axis=1)
    x_test_comb = pd.concat([x_test_cat_with_missing_
    values_encoded,
    x_test_num_with_missing], axis=1)
    return (x_train_comb, x_test_comb)

def train_test_evalute(x_train, y_train, x_test, y_test,
    n_estimators = [3, 6, 50, 100, 200],
    max_depth=[3, 6, 10, 50, 100],
    criterion=["entropy", "gini"]):
    results = {}
    best_auc = 0
    best_params = None

    for crit in criterion:
        accuracy_by_estimator_count = {}

        for estimator_count in n_estimators:
            accuracy_by_depth = {}

            for depth in max_depth:
                # create algorithm
                random_forest_algo = RandomForestClassifier
                (criterion=crit, class_weight='balanced',
                 n_jobs=-1, n_estimators=estimator_count,
                 max_depth=depth, random_state=42)
                # train
                random_forest_algo.fit(x_train, y_train)
                # predict test dataset
                y_pred = random_forest_algo.predict(x_test)
                y_pred_proba1 = random_forest_algo.predict_proba
                (x_test)[:,1]
                # calculate accuracy
                accuracy = accuracy_score(y_test, y_pred)
                roc_auc = roc_auc_score(y_test, y_pred_proba1)
```

```
                # save best accuracy and parameters
                # save best roc_auc score

                if roc_auc > best_auc:
                    best_auc = roc_auc
                    best_params = {
                        'n_estimators': estimator_count,
                        'max_depth': depth,
                        'criterion': crit
                    }
#                   if accuracy > best_accuracy_score:
#                       best_accuracy_score = accuracy
#                       best_params = {
#                           'n_estimators': estimator_count,
#                           'max_depth': depth,
#                           'criterion': crit
#                       }
                accuracy_by_depth[depth] = {
                    "accuracy":accuracy,
                    "roc_auc":roc_auc
                }
            accuracy_by_estimator_count[estimator_count] =
        accuracy_by_depth
        results[crit] = accuracy_by_estimator_count
    return best_params, best_auc, results

# read dataset
data = pd.read_csv("train.csv")

# get "TARGET" column into y
target = data.pop('TARGET')
y = target.values

# Remove identifers from features
data.pop('SK_ID_CURR')

# split train and test data
x_train, x_test, y_train, y_test = train_test_split
(data, y, stratify=y, test_size=0.33, random_state=42)
```

```
# preprocess data to be suitable for training
x_train_proc, x_test_proc = preprocess_dataset
(x_train, x_test)

best_algorithm_parameters, best_auc, all_results =
train_test_evalute(x_train_proc, y_train, x_test_proc, y_test)

print(best_auc)
print(best_algorithm_parameters)

# Training best classifier
random_forest_algo = RandomForestClassifier
(criterion=best_algorithm_parameters['criterion'],
 class_weight='balanced', n_estimators=
 best_algorithm_parameters['n_estimators'],
 max_depth=best_algorithm_parameters['max_depth'],
 random_state=42)
random_forest_algo.fit(x_train_proc, y_train)
y_pred = random_forest_algo.predict(x_test_proc)

# confusion matrix
confusion_matrix(y_test, y_pred)

#accuracy
accuracy = accuracy_score(y_test,y_pred)
print(accuracy)
```

6.2 Prikaz podataka

Učitavanje podataka

```
import seaborn as sns
sns.set(rc={'figure.figsize':(11.7,8.27)}, style="darkgrid")
data = pd.read_csv("train.csv")
```

Slika 5.2 - Deset najznačajnijih značajki

```
# Get most important features
feature_importance = random_forest_algo.feature_importances_
feature_indices = feature_importance.argsort()[::-1]

top_features = feature_indices[:10]
top_feature_names = [x_train_proc.columns[feature_index]
for feature_index in top_features]
top_feature_weights = [feature_importance[feature_index]
for feature_index in top_features]

ax = sns.barplot(y=np.array(top_10_feature_names),
x=np.array(top_feature_importance), orient="h");
ax.set(title="Tezine 10 najznacajnihij znacajki",
xlabel="Tezina znacajke", ylabel="Znacajka")
```

Slika 5.3 - Distribucija godina

```
birth_years = data["DAYS_BIRTH"]/-365
ax = sns.distplot(birth_years,
bins=10, kde=False)
ax.set(title="Distribucija godina",
xlabel='Broj godina', ylabel='Broj osoba')
```

Slika 5.4 - Duljina radnog staža dužnika

```
correct_data = data["DAYS_EMPLOYED"].replace({365243: np.nan})
# Pogreska u podatcima, imamo osobe zaposlene 1000 godina
correct_data_as_year =
correct_data/-365 # convert to number of years employed
ax = sns.distplot(correct_data_as_year, bins=50, kde=False)
ax.set(title="Distribucija DAYS_EMPLOYED",
xlabel='Broj godina zaposlenja', ylabel='Broj osoba')
```

Slika 5.5 - Razina obrazovanja dužnika

```
ax = sns.countplot(x="TARGET",
hue="NAME_EDUCATION_TYPE", data=data);
ax.set(xlabel='Razina obrazovanja', ylabel='Broj osoba')
ax.legend(['sekundarno', 'Visoko obrazovanje',
'nepotpuno obrazovanje', 'nize sekundarno', 'akademsko'])
ax.set_xticklabels(['Dug je otplacen', 'Dug nije otplacen'])
```

Bibliografija

[1] L. Breiman, *Random Forests*, University of California, 2001.

[2] Šnajder J. Dalbelo Bašić, B., *Strojno učenje*, (2017), https://www.fer.unizg.hr/_download/repository/SU-2019-01-Uvod.pdf.

[3] A. Field, *Discovering Statistics Using SPSS*, SAGE Publications, 2009.

[4] Home Credit Group, *Home Credit Default Risk*, (2018), https://www.kaggle.com/c/home-credit-default-risk/data.

[5] S. Hartshorn, *Machine Learning With Random Forests And Decision Trees*, Kindle edition, 2016.

[6] Tibshirani R. Friedman J. Hastie, T., *The Elements of Statistical Learning*, Springer, 2009.

[7] Lemeshow S. Hosmer, D.W., *Applied Logistic Regression*, John Wiley and sons, 2000.

[8] Punchihewa H. Emile J. Morrison J. Mayo, H., *Strojno učenje*, (2018), https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html.

[9] S.J. Miller, *The Method of Least Squares*, Brown University, 2006.

[10] T.M. Mitchell, *Machine learning*, McGraw-Hill Science, 1997.

[11] Peck E.A. Vining G.G. Montgomery, D.C., *Introduction to Linear Regression Analysis*, Wiley, 2012.

[12] nepoznato, *6. tematska jedinica*, (2013), https://fmtu.lumens5plus.com/sites/fmtu.lumens5plus.com/files/104-5dbdec8cba4bbd2198d6b3f25fa297c2.pdf.

[13] Ž. Pauše, *Uvod u matematičku statistiku*, Školska knjiga, 1993.

[14] Vondraček Z. Sandrić, N., *Vjerojatnost*, PMF Zagreb, 2019.

[15] S. Singer, *Učenje na primjerima*, (2020), http://degiorgi.math.hr/~singer/ui/ui_1415/ch_18b.pdf,.

[16] V. Čulajk, *Vjerojatnost i statistika*, Građevinski fakultet, 2011.

[17] Merćep A. Đuričić, T., *Strojno učenje - bilješke sa predavanja*, (2015), https://www.fer.unizg.hr/_download/repository/SU-2015-Vrednovanje_modela.pdf.

[18] J. Šnajder, *Linearan model regresije*, (2017), https://www.fer.unizg.hr/_download/repository/SU-Regresija.pdf.

Sažetak

U ovome radu glavni predmet proučavanja bila je primjena modela slučajnih šuma u svakodnevnom života, točnije u računanju kreditnog rizika u bankama. U početnom dijelu rada pobliže smo upoznali čitatelja s jednostavnom i višestrukom linearnom regresijom. U nastavku rada donijeli smo pregled razvoja strojnog učenja te pojasnili istoimeni pojam. Zatim smo detaljno opisali pojam stabla odluke i kako se ono razvija te smo njegov model prikazali na primjeru i vizualno radi lakšeg shvaćanja napisanog. Slijedilo je poglavlje posvećeno teorijskoj obradi modela slučajnih šuma i njihovoj primjeni u bankarstvu. Radi lakše interpretacije podaci su prikazani kroz histograme i tablice. Za kraj rada donesi su zaključci o radu algoritma i mogućnosti njegove implementacije u bankarski sustav.

Ključne riječi: linearna regresija, strojno učenje, stablo odluke, slučajna šuma

Summary

In this paper, the main subject of study was application of the model of random forests in everyday life, more precisely in the calculation of credit risk in banks. In the initial part of the paper, we introduced the reader to simple and multiple linear regression. In the continuation of the paper, we provide an overview of the development of machine learning and clarify the concept of them. Then we minutely described the concept of the decision tree and how it develops, and we presented its model on an example and visually to make it easier to understand what is written. This was followed by a chapter dedicated to the theoretical processing of random forest models and their application in banking. For easier interpretation, the data are presented through histograms and tables. At the end of the paper, conclusions were made about the operation of the algorithm and the possibility of its implementation in the banking system.

Key words: linear regression, machine learning, decision tree, random forest

Životopis

Rođen sam 25.11.1995. u Ottawi, glavnom gradu Kanade. Pohađam OŠ Bartola Kašića i XI. gimnaziju u Zagrebu. 2014. godine upisujem preddiplomski sveučilišni studij Matematika; smjer nastavnički na Prirodoslovno-matematičkom fakultetu u Zagrebu. Preddiplomski studij završavam 2018. godine te akademske godine 2018./2019. upisujem diplomski sveučilišni studij Matematika; smjer nastavnički, koji završavam 2020. godine. Tijekom studiranja dobivam Rektorovu nagradu za znanstveni rad 2017., Posebno rektorovo priznanje za postignuća u sportu 2016., Nagradu matematičkog odsjeka za iznimne rezultate 2019. te Nagradu PMF-a za sportske uspjehe 2018. Uz akademske aktivnosti bavim se sportom. Nakon deset godina igranja u Hrvatskom akademskom vaterpolskom klubu Mladost, osvojenih pet naslova prvaka države i pregršt domaćih inozemnih medalja, 2017. godine prelazim u WBV Graz gdje igram i danas. Vrijedi istaknuti srebrnu medalju na Europskim sveučilišnim igrama 2016. i titulu najboljeg strijelca Bundeslige (1. austrijske lige) u sezoni 2017./2018.