UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

MAGISTRSKO DELO

Vito Čoh

Maribor, 2020

UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

Magistrsko delo

UPORABA STROJNEGA UČENJA ZA NAPOVEDOVANJE ŠKODNIH DOGODKOV

na študijskem programu 2. stopnje Matematika

Mentor:

izr. prof. dr. Marko Jakovac

Somentor:

Bor Harej

Kandidat:

Vito Čoh

Maribor, 2020

ZAHVALA

Zahvaljujem se mentorju izr. prof. dr. Marku Jakovcu za odzivnost, nasvete in strokovno usmerjanje pri nastajanju tega magistrskega dela. Prav tako se zahvaljujem somentorju Boru Hareju za odzivnost in strokovno usmerjanje s področja strojnega učenja pri neživljenjskih zavarovanjih.

Posebej se zahvaljujem obema staršema, ki sta me na moji poti vedno podpirala. Zahvaljujem se tudi sestri in vsem ostalim, ki so del mojega življenja.

Vsem iskreno hvala!

Priloga 6 – IZJAVA O AVTORSTVU IN ISTOVETNOSTI TISKANE IN ELEKTRONSKE OBLIKE ZAKLJUČNEGA DELA

UNIVERZA V MARIBORU
Fakulteta za naravoslovje in matematiko
(ime članice UM)

IZJAVA O AVTORSTVU IN ISTOVETNOSTI TISKANE IN ELEKTRONSKE OBLIKE ZAKLJUČNEGA DELA

Ime in priimek študent-a/-ke: Vito Čoh
Študijski program: MATEMATIKA
Naslov zaključnega dela: Uporaba strojnega učenja za napovedovanje škodnih dogodkov

Mentor: Marko Jakovac
Somentor: Bor Harej

Podpisan-i/-a študent/-ka Vito Čoh

- izjavljam, da je zaključno delo rezultat mojega samostojnega dela, ki sem ga izdelal/-a ob pomoči mentor-ja/-ice oz. somentor-ja/-ice;
- izjavljam, da sem pridobil/-a vsa potrebna soglasja za uporabo podatkov in avtorskih del v zaključnem delu in jih v zaključnem delu jasno in ustrezno označil/-a;
- na Univerzo v Mariboru neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico ponuditi zaključno delo javnosti na svetovnem spletu preko DKUM; sem seznanjen/-a, da bodo dela deponirana/objavljena v DKUM dostopna široki javnosti pod pogoji licence Creative Commons BY-NC-ND, kar vključuje tudi avtomatizirano indeksiranje preko spleta in obdelavo besedil za potrebe tekstovnega in podatkovnega rudarjenja in ekstrakcije znanja iz vsebin; uporabnikom se dovoli reproduciranje brez predelave avtorskega dela, distribuiranje, dajanje v najem in priobčitev javnosti samega izvirnega avtorskega dela, in sicer pod pogojem, da navedejo avtorja in da ne gre za komercialno uporabo;
- dovoljujem objavo svojih osebnih podatkov, ki so navedeni v zaključnem delu in tej izjavi, skupaj z objavo zaključnega dela;
- izjavljam, da je tiskana oblika zaključnega dela istovetna elektronski obliki zaključnega dela, ki sem jo oddal/-a za objavo v DKUM.

Uveljavljam permisivnejšo obliko licence Creative Commons: _____ (navedite obliko)

Datum in kraj: Maribor, 07.02.2020 Podpis študent-a/-ke:

_____

Uporaba strojnega učenja za napovedovanje škodnih dogodkov

program magistrskega dela

Na področju neživljenjskih škod praviloma uporabljamo že uveljavljene metode, ki so bile razvite v času, ko še niso bili na voljo sodobni računalniki. V primeru škodnih rezervacij lahko takšne škode ponazorimo s pomočjo trikotnikov razvoja. Gre za razporeditev podatkov o škodah v tabele glede na leta nastanka in leta razvoja rešitve, zato trikotnike razvoja uporabljamo za ocenjevanje morebitnih prihodnjih škod na podlagi preteklih škod, ki so že bile prijavljene. Primer takšne metode, ki sloni na trikotniku razvoja, je metoda veriženja (chain-ladder). Zaradi danes razpoložljive računalniške moči, je izguba informacij, ki nastane pri združevanju podatkov glede na leta nastanka in leta razvoja, komajda upravičena, zato bodo v magistrskem delu predstavljeni primeri uporabe strojnega učenja pri napovedovanju škodnih zahtevkov.

Osnovni viri:

1. B. Harej, R. Gächter, S. Jamal, *Individual Claim Development with Machine Learning*, ASTIN working party report, 2017.

2. S. Jamal, S. Canto, R. Fernwood, C. Giancaterino, M. Hiabu, L. Invernizzi, T. Korzhynska, Z. Martin, H. Shen, *Machine Learning and Traditional Methods Synergy in Non-Life Reserving*, ASTIN working party report, 2018.

3. H.I. Witten, E. Frank, A.M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3. edition, Morgan Kaufmann Publishers, 2011.

izr. prof. dr. Marko Jakovac

Bor Harej

ČOH, V.: Uporaba strojnega učenja za napovedovanje škodnih dogodkov. Magistrsko delo, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Oddelek za matematiko in računalništvo, 2020.

IZVLEČEK

V magistrskem delu je predstavljena uporaba posplošenega linearnega modela in različnih metod strojnega učenja v zavarovalništvu. Delo je razdeljeno na teoretični in praktični del.

Na začetku teoretičnega dela so opisani osnovni pojmi iz verjetnosti in zavarovalništva. Predstavljeno je tudi, kako zavarovalnice določijo višino premije. Nato sta predstavljena teoretično ozadje posplošenega linearnega modela in uporaba tega modela za napovedovanje višine škode. Na koncu teoretičnega dela pa je opisano strojno učenje in bolj podrobno so predstavljena odločitvena drevesa, naključni gozdovi ter nevronske mreže.

V praktičnem delu magistrskega dela pa so posplošeni linearni model, naključni gozd in nevronska mreža uporabljeni za napovedovanje višine škode pri avtomobilskem zavarovanju. Najprej so podatki predstavljeni ter ustrezno obdelani. Nato so določeni parametri posameznih modelov. Na koncu pa so modeli med seboj primerjani in izbran je najboljši model.

Ključne besede: zavarovalništvo, posplošeni linearni model, strojno učenje, odločitveno drevo, naključni gozd, nevronska mreža.

Math. Subj. Class. (2010): 62J12, 62P05, 97M30.

ČOH, V.: Using machine learning to predict loss events.
Master Thesis, University of Maribor, Faculty of Natural Sciences and Mathematics, Department of Mathematics and Computer Science, 2020.

ABSTRACT

The master thesis presents the use of generalized linear model and different machine learning methods in insurance. It is divided into theoretical and practical part.

At the beginning of the theoretical part, basic notions of probability and insurance are described. It is also presented, how insurance companies determine the insurance premium. Then the theoretical background of generalized linear model and the use of it in predicting claim amount are presented. At the end of the theoretical part, machine learning is described and also decision trees, random forests and neural networks are presented in detail.

The practical part of master thesis is focused on how generalized linear model, random forest and neural network are used for predicting car insurance claims. First, the data is presented and processed. Then the parameters of each model are determined. In the end, the models are compared and the best one is chosen.

Keywords: insurance, generalized linear model, machine learning, decision tree, random forest, neural network.

Math. Subj. Class. (2010): 62J12, 62P05, 97M30.

Kazalo

Uvod

Med zavarovalnicami na trgu vlada velika konkurenca, zato je za posamezno zavarovalnico pomembno, kako dobro je zmožna prepoznati nerizične in rizične zavarovance. To pomeni, da mora zavarovalnica razviti metodo, s katero bo identificirala lastnosti manj rizičnih zavarovancev, katerim bo zaračunala nižjo premijo kot konkurenca, in lastnosti bolj rizičnih zavarovancev, katerim pa bo zaračunala višjo premijo kot konkurenca, saj se jim želi izogniti. Torej je za zavarovalnico ključno, da prepozna manj rizične zavarovance in jim ponudi nižjo premijo, kot zavarovancem, ki spadajo v bolj rizično skupino. To dejansko pomeni, da zavarovancem določi višino premije na podlagi njihovih lastnosti.

Na evropskem trgu se že dalj časa za določanje premij pri avtomobilskem zavarovanju uporablja posplošeni linearni model. Z njegovo pomočjo lahko identificiramo lastnosti posameznika in vozila, ki vplivajo na rizičnost, hkrati pa lahko ugotovimo, v kolikšni meri posamezna lastnost vpliva na rizičnost ter posledično na višino premije.

Zaradi velikega napredka v računalniški moči pa se je v zadnjih desetletjih močno razvilo strojno učenje, zato se pojavita naslednji vprašanji:

- Ali lahko strojno učenje uporabimo za prepoznavanje rizičnih lastnosti zavarovancev in določitev premije?
- Ali lahko s strojnim učenjem dobimo boljše rezultate kot s posplošenim linearnim modelom?

V tem magistrskem delu, ki je razdeljeno na teoretični in praktični del, smo poskušali odgovoriti na ti dve vprašanji. V teoretičnem delu smo najprej razložili osnovne pojme iz verjetnosti in zavarovalništva, ki se pojavljajo v nadaljevanju. Nato smo predstavili posplošeni linearni model in različne metode strojnega učenja, med katere spadata tudi naključni gozd in nevronska mreža. V praktičnem delu pa smo različne modele uporabili za napovedovanje višine škode pri avtomobilskih zavarovanjih. Med seboj smo primerjali posplošeni linearni model, naključni gozd in nevronsko mrežo ter na koncu določili najboljši model.

Poglavje 1

Osnovni pojmi

V tem poglavju so predstavljeni osnovni pojmi iz statistike in verjetnosti, ki se pojavijo v nadaljevanju magistrske naloge. Vsi pojmi so povzeti iz [9].

Verjetnostni račun je sestavljen iz poskusov, dogodkov in verjetnosti dogodkov. Poskus je dejavnost, ki jo izvedemo pod natančno določenimi pogoji, dogodek pa je pojav, ki ga opazujemo pri poskusu. Nemogoč dogodek, ki ga označimo z N, je dogodek, ki se ne zgodi v nobeni ponovitvi poskusa. Gotov dogodek, ki ga označimo z G, je dogodek, ki se zgodi v vseh ponovitvah poskusa. Naključni dogodek pa je dogodek, ki se v nekaterih ponovitvah poskusa zgodi, v drugih pa ne.

Definicija 1.1 *Neprazno množico* $G = \{e_\lambda \mid \lambda \in \Lambda\}$ *imenujemo prostor elementarnih dogodkov. Poljubno podmnožico* $A \subseteq G$ *imenujemo dogodek, njegov nasprotni dogodek pa označimo z* $\bar{A}$.

Definicija 1.2 *Naj bosta* $A, B \subseteq G$ *poljubna dogodka.*

- *Vsota dogodkov* $A \cup B$ *se zgodi, ko nastopi dogodek* A *ali dogodek* B.
- *Produkt dogodkov* AB *se zgodi, ko hkrati nastopita dogodka* A *in* B.
- *Dogodka* A *in* B *sta nezdružljiva, če se ne moreta zgoditi hkrati, to pomeni* $AB = N$.

Definicija 1.3 *Naj bo* G *prostor elementarnih dogodkov. Neprazna družina podmnožic* $\mathcal{A}$ *v* G *je* σ*-algebra, če velja*

- $A \in \mathcal{A} \Rightarrow \bar{A} \in \mathcal{A}$,

- *za vsako družino* $\{A_n \,|\, n \in \mathbb{N}\} \subseteq \mathcal{A}$ *je* $\bigcup\limits_{n\in\mathbb{N}} A_n \in \mathcal{A}$.

Borelova σ-algebra na $\mathbb{R}$, ki jo označimo z $\mathcal{B}$, je najmanjša σ-algebra na $\mathbb{R}$, ki vsebuje vse odprte množice. Njene elemente imenujemo Borelove množice.

Definicija 1.4 *Verjetnost na* σ*-algebri* $\mathcal{A}$ *je preslikava* $P : \mathcal{A} \to \mathbb{R}$*, ki zadošča naslednjim aksiomom:*

- $P(A) \geq 0, \quad \forall A \in \mathcal{A}$,
- $P(G) = 1$,
- *za vsako družino* $\{A_n \mid n \in \mathbb{N}\} \subseteq \mathcal{A}$ *paroma nezdružljivih dogodkov velja*

$$P\left(\bigcup_{n\in\mathbb{N}} A_n\right) = \sum_{n\in\mathbb{N}} P(A_n).$$

Število $P(A)$ imenujemo verjetnost dogodka A, urejeno trojico $(G, \mathcal{A}, P)$ pa imenujemo verjetnostni prostor.

Definicija 1.5 *Naj bosta* $A, B \subseteq G$ *poljubna dogodka. Dogodka* A *in* B *sta neodvisna, če je* $P(AB) = P(A)P(B)$.

Definicija 1.6 *Naj bo* $(G, \mathcal{A}, P)$ *verjetnostni prostor. Funkcija* $X : G \to \mathbb{R}$ *je naključna spremenljivka, če za vsako Borelovo množico* $B \in \mathcal{B}$ *velja*

$$X^{-1}(B) = \{e \in G \mid X(e) \in B\} \in \mathcal{A}.$$

Definicija 1.7 *Naj bo* X *naključna spremenljivka. Porazdelitvena funkcija naključne spremenljivke* X*,* $F_X : \mathbb{R} \to [0, 1]$*, je definirana s predpisom*

$$F_X(x) = P(X < x).$$

Poznamo dva tipa naključnih spremenljivk glede na zalogo vrednosti.

Definicija 1.8 *Naključna spremenljivka* X *je diskretna, če je njena zaloga vrednosti števna.*

Definicija 1.9 *Naključna spremenljivka X je zvezna, če obstaja nenegativna funkcija $p : \mathbb{R} \to \mathbb{R}$, da velja*

$$F_X(x) = P(X < x) = \int_{-\infty}^{x} p(t)dt, \qquad \forall x \in \mathbb{R}.$$

Funkcijo p imenujemo gostota verjetnosti naključne spremenljivke X.

Definicija 1.10 (Matematično upanje)

- *Naj bo X diskretna naključna spremenljivka z verjetnostno funkcijo $p_n = P(X = x_n)$, za vsak n. Če vrsta $\sum_n |x_n| p_n$ konvergira, potem izraz*

$$E(X) = \sum_n p_n x_n$$

  imenujemo matematično upanje diskretne naključne spremenljivke X.

- *Naj bo X zvezna naključna spremenljivka z gostoto verjetnosti p. Če integral $\int_{\mathbb{R}} |x| p(x)dx$ konvergira, potem izraz*

$$E(X) = \int_{\mathbb{R}} x\, p(x)dx$$

  imenujemo matematično upanje zvezne naključne spremenljivke X.

Matematično upanje ali povprečna vrednost je mera za osrednjo tendenco porazdelitve.

Definicija 1.11 (Disperzija)
Naj bo X naključna spremenljivka, za katero obstaja matematično upanje $E(X)$. Njena disperzija je definirana s predpisom

$$Var(X) = E\left((X - E(X))^2\right).$$

Disperzija ali varianca je mera za razpršenost porazdelitve okoli njene povprečne vrednosti. Izraz $\sigma(X) = \sqrt{Var(X)}$ pa imenujemo standardni odklon.

Definicija 1.12 *Naj bo naključna spremenljivka X porazdeljena po gama porazdelitvi $\Gamma(\alpha, \beta)$. Pri tem sta $\alpha, \beta > 0$. Gostota verjetnosti od X je*

$$p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \qquad x > 0,$$

kjer je $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$ Eulerjeva funkcija gama.
Za naključno spremenljivko X velja, da je $E(X) = \frac{\alpha}{\beta}$ in $Var(X) = \frac{\alpha}{\beta^2}$.

Poglavje 2

Zavarovalništvo

V tem poglavju so predstavljeni in opisani pojmi povezani z zavarovalništvom.

2.1 Zavarovanje

Za definicijo zavarovanja največkrat uporabimo opredelitev, ki se nahaja v knjigi [1]: "Zavarovanje je ustvarjanje gospodarske varnosti z izravnavanjem gospodarskih nevarnosti. To je očitno gospodarska dejavnost, katere cilj je ustvarjanje gospodarske varnosti in sredstvo za izravnavanje nevarnosti." Dr. Boncelj v tej knjigi zapiše tudi: "Zavarovanje je edina gospodarska dejavnost, katere bistvo izvira iz uveljavljanja zakona velikih števil."

Princip zavarovanja je torej, da se številna tveganja, katerim so izpostavljeni zavarovanci, prerazporedijo na vse zavarovance, ter da se zavarovancu izplača ustrezno nadomestilo za utrpelo škodo v skladu s sklenjeno zavarovalno pogodbo. Za posameznika nikoli ne moremo trditi, ali bo, in kdaj bo izpostavljen uresničitvi nekega škodnega dogodka, za veliko skupino posameznikov pa pojav določenih škodnih dogodkov lahko predvidimo z veliko verjetnostjo [3].

Posamezniki se zavarujemo predvsem zato, ker s sklenitvijo zavarovanja neko negotovost zamenjamo z gotovostjo. Na tak način se zavarujemo pred prihodnjim negotovim dogodkom, ki nam lahko povzroči veliko škode. S sklenjenim zavarovanjem, nam to škodo v določeni meri poplača zavarovalnica. Tako negotov dogodek spremenimo v gotov, saj škode ne krijemo sami [4].

Zavarovalni dogodek je dogodek, ki nastane zaradi nevarnosti, ki jo zavarovalna pogodba krije in zaradi katere nastane škoda na zavarovanem predmetu.

2.2 Premija

Zavarovalna premija je znesek, ki ga zavarovanec plača zavarovalnici ob sklenitvi zavarovanja v zameno za zavarovalno kritje [2]. Ta znesek imenujemo tudi kosmata ali bruto premija. Ker je ta znesek odvisen od stroškov, velikosti tveganja in drugih dejavnikov, kosmato premijo delimo na čisto in stroškovno premijo.

Čista premija zagotavlja sredstva za uresničevanje funkcij zavarovanja, zato se zanjo uporablja tudi izraz funkcionalna premija [2]. Ta se nato deli na tehnično premijo in varnostni dodatek. Tehnična premija zagotavlja, da lahko ob nastali škodi zavarovalnica le-to poplača. Biti mora dovolj velika, da pokrije vso škodo, ki jo zavarovanec utrpi. Tehnična premija se naprej loči na nevarnostno in varčevalno premijo, ki pa je značilna za življenjska zavarovanja. Nevarnostna premija je del tehnične premije, ki v povprečju zadošča za kritje vseh škod iz sprejetih tveganj. Določi se s pomočjo statistične obravnave preteklega škodnega dogajanja. Zaradi možnih negativnih odstopanj od pričakovanj, pa je potreben prej omenjeni varnostni dodatek, ki pokriva ta odstopanja.

Stroškovna premija se teoretično določa glede na razmejene stroške za posamezno zavarovanje [2]. Sem sodijo vsi stroški, ki so povezani z izdajo zavarovanja, obdelavo podatkov, marketingom itd.

Ključni del pri določanju premije je torej izračun nevarnostne premije, saj mora ta pokriti škode, ki bodo nastale iz sprejetih tveganj. To premijo zavarovalnica izračuna vnaprej. Ker v trenutku izračuna ne ve, kakšno bo posameznikovo škodno dogajanje, mora nevarnostno premijo izračunati na podlagi preteklega škodnega dogajanja s pomočjo statističnih analiz. V praksi lahko nevarnostno premijo določimo kot razmerje med izplačanimi škodami in ustrezno enoto izpostavljenosti oziroma kot produkt povprečne škode in škodne pogostosti. Enota izpostavljenosti je merska enota, s katero izmerimo število enakovrstnih rizikov, ki so zavarovani v nekem časovnem intervalu [5]. Za boljše razumevanje si poglejmo naslednji primer.

Zavarovalnica je v preteklosti sprejela n enoletnih tveganj (enoletne police). Izplačala je k škod v višini $x_1, x_2, \ldots, x_k$. Celoten znesek, ki ga je zavarovalnica izplačala, je torej enak vsoti

$$S_n = \sum_{i=1}^{k} x_i.$$

Na podlagi zgornjih podatkov lahko izračunamo povprečno škodo, škodno pogostost ter nevarnostno premijo:

- Povprečna škoda (ps) - povprečno izplačana škoda:

$$ps = \frac{S_n}{k}.$$

- Škodna pogostost (sp) - število škod na enoto izpostavljenosti:

$$sp = \frac{k}{n}.$$

- Nevarnostna premija (np) - produkt povprečne škode in škodne pogostosti oziroma razmerje med izplačanimi škodami in enoto izpostavljenosti:

$$np = ps \cdot sp = \frac{S_n}{k} \cdot \frac{k}{n} = \frac{S_n}{n}.$$

Ta pristop za izračun nevarnostne premije lahko še razvijemo. Možen način je posplošeni linearni model, ki ga bomo predstavili v nadaljevanju. Na koncu pa bomo posplošeni linearni model poskušali nadomestiti s strojnim učenjem in nato primerjali rezultate.

2.3 Določanje cen v zavarovalništvu

Pri določanju cen v zavarovalništvu je pomembno, da zavarovalnica zmanjša tveganje. Običajno to naredi tako, da poveča število zavarovancev ter s tem zmanjša varianco povprečnega tveganja, ki ga je sprejela v zavarovanje. Na ta način lahko zavarovalnica bolj natančno oceni pričakovano škodo, saj se z večanjem števila zavarovanj odkloni od povprečja manjšajo. Nevarnostna premija je zato enaka pričakovanim škodam, ki so posledica tveganj, sprejetih v zavarovanje [4]. O tem govori tudi zakon velikih števil oziroma centralni limitni izrek.

Izrek 2.1 (Centralni limitni izrek)
Naj bodo $X_1, X_2, \ldots, X_n$ enako porazdeljene, paroma neodvisne naključne spremenljivke z matematičnim upanjem μ in disperzijo σ^2. Potem velja, da se vsota porazdelitev $S_n = X_1 + X_2 + \cdots + X_n$ približuje normalni porazdelitvi, ko gre $n \to \infty$:

$$S_n \xrightarrow{n\to\infty} N(n\mu, n\sigma^2),$$
$$\frac{S_n}{n} \xrightarrow{n\to\infty} N(\mu, \frac{\sigma^2}{n}).$$

Pri oblikovanju cen v zavarovalništvu je pomembno tudi, kako dobro prepoznamo različne rizične skupine in definiramo, v katero skupino spada določen posameznik. Različne rizične

skupine se na primer pojavijo pri avtomobilskem zavarovanju, kjer lahko predpostavimo, da so mladi vozniki, tj. vozniki z manj kot tremi leti izkušenj, bolj rizična skupina. Če zavarovalnica pri določanju cen ne upošteva te lastnosti, pride do tega, da zavarovanci z več let izkušenj, ki so po predpostavki manj rizična skupina, plačujejo višjo premijo, saj se višina premije kompenzira z manj izkušenimi vozniki, ki plačujejo nižjo premijo, kot bi jo sicer.

Problem se pojavi, ko je na trgu večje število zavarovalnic. Če ena zavarovalnica določa cene tako, da upošteva različne rizične skupine, druga zavarovalnica pa tega ne stori, potem slednja hitro postane nekonkurenčna za manj rizično skupino zavarovancev in pridobi samo bolj rizično skupino zavarovancev, kar pa si verjetno ne želi.

Rizičnost skupine lahko določimo na podlagi različnih lastnosti oziroma spremenljivk. To so lahko:

- osebne lastnosti posameznika (vozniške izkušnje, starost zavarovanca, spol, ...),
- lastnosti zavarovančevega objekta (moč in starost vozila, vrsta gradnje hiše, ...),
- lastnosti kraja (bližina vode, število prebivalcev, bližina hribovja, ...).

Tudi tukaj se pojavijo problemi. Včasih ne moremo pridobiti podatkov, ki bi dobro opisali razlike med rizičnimi skupinami, ali pa so ti podatki neresnični in težko preverljivi. Lahko se tudi zgodi, da nekaterih spremenljivk ne smemo upoštevati. Ena takšnih spremenljivk je na primer spol, ki ga zaradi diskriminacije ne smemo upoštevati kot lastnost neke rizične skupine.

2.4 Multiplikativni model

Kot je opisano v podpoglavju 2.3, je ključno, da zavarovance na podlagi njihovih lastnosti razdelimo v razrede. To pomeni, da zavarovance z istimi lastnostmi razvrstimo v isti razred in jim znotraj posameznega razreda zaračunamo isto premijo. Temu pravimo tudi, da zavarovanci sodijo v isti premijski razred. Če imamo v vsakem razredu na voljo dovolj podatkov o škodah, lahko dokaj preprosto izračunamo premijo. Ta je po centralnem limitnem izreku enaka pričakovani škodi zavarovancev v tem premijskem razredu.

Problem se pojavi, ko imamo opravka z velikim številom različnih razredov. Recimo, da imamo p različnih spremenljivk oziroma lastnosti in vsaka spremenljivka ima k_i razredov, $i = 1, \ldots, p$. Potem je število vseh premijskih razredov enako $\prod_{i=1}^{p} k_i$, kar pomeni, da se število premijskih razredov hitro povečuje s številom spremenljivk. Posledično se lahko zgodi, da

bomo v nekaterih razredih imeli zelo malo podatkov in bo zato varianca v teh razredih zelo velika. Rešitev tega problema je multiplikativni model, pri katerem predpostavimo, da lastnosti zavarovanca multiplikativno vplivajo na njegovo rizičnost. To metodo lahko uporabimo pri modeliranju prej omenjenih vrednostih: škodne pogostosti, povprečne škode in nevarnostne premije.

V nadaljevanju bomo predstavili multiplikativni model. Naj i predstavlja i-ti premijski razred in μ_i vrednost, ki jo želimo modelirati. To je lahko povprečna škoda, škodna pogostost ali nevarnostna premija. Naš model naj sestavlja p različnih faktorjev oziroma spremenljivk. Torej lahko i-ti premijski razred zapišemo kot $i = (i_1, i_2, \ldots, i_p)$, kjer i_j označuje nivo j-tega faktorja v i-tem premijskem razredu. Pri multiplikativnem modelu predpostavljamo, da lahko μ_i zapišemo kot

$$\mu_i = \gamma_0 \gamma_{1i_1} \gamma_{2i_2} \cdots \gamma_{pi_p},$$

kjer γ_{ji_j}, $j = 1, \ldots, p$, določa, koliko se rizičnost zavarovanca poveča ali zmanjša na podlagi j-tega faktorja, če je nivo tega faktorja enak i_j [5].

Omenimo še, da multiplikativni model upošteva korelacije med faktorji oziroma spremenljivkami, ne pa tudi interakcije. To pomeni, da ne upošteva odvisnosti faktorjev med seboj.

Parametre γ_{ji_j}, ki se pojavijo v multiplikativnem modelu, lahko ocenimo s pomočjo posplošenega linearnega modela, zato si ta model poglejmo bolj podrobno.

Poglavje 3

Posplošeni linearni model

Posplošeni linearni model predstavlja razširitev linearnega modela, ki je definiran s predpisom

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n + \epsilon,$$

kjer so $X_1, \ldots, X_n$ pojasnjevalne spremenljivke, n je število pojasnjevalnih spremenljivk, Y je odvisna spremenljivka in ϵ je napaka, ki je porazdeljena $\epsilon \sim N(0, \sigma^2)$.

Osnove posplošenega linearnega modela oziroma GLM (angl. Generalized Linear Model) sta postavila J. Nelder in R. Wedderburn leta 1972.

GLM je sestavljen iz treh komponent [8]:

1. Naključna komponenta.
   Nanaša se na porazdelitev odvisne spremenljivke Y.

2. Sistematična komponenta.
   Linearni prediktor η je linearna kombinacija pojasnjevalnih spremenljivk $X_1, \ldots, X_n$:

   $$\eta = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n.$$

3. Povezovalna funkcija.
   Povezava med naključno in sistematično komponento je opisana s povezovalno funkcijo g, tako da za pričakovano vrednost odvisne spremenljivke Y velja

   $$E(Y) = g^{-1}(\eta).$$

Posledično lahko GLM interpretiramo kot nelinearni regresijski model za odvisno spremenljivko. V nadaljevanju bomo posamezne komponente predstavili bolj podrobno.

3.1 Naključna komponenta

Komponente naključnega vektorja $Y = (Y_1, Y_2, \ldots, Y_m)$ so neodvisne naključne spremenljivke, ki pripadajo družini eksponentno porazdeljenih naključnih spremenljivk. Gostota verjetnosti za to družino je oblike [5]

$$p_{Y_i}(y_i; \theta_i, \phi) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{\phi/\omega_i} + c(\phi, \omega_i; y_i)\right). \tag{3.1}$$

Pri tem je $\phi > 0$ parameter disperzije in je enak za vse komponente od Y, parameter θ_i je lahko za vsako komponento od Y različen, ω_i pa predstavlja utež, s katero lahko i-ti izid predhodno ustrezno utežimo. Funkcija b je dvakrat odvedljiva v $\mathbb{R}$, funkcija c pa je dvakrat odvedljiva v $\mathbb{R}^2$. Funkcijo b imenujemo tudi kumulativna funkcija in z njeno izbiro določimo tip porazdelitve iz družine eksponentnih porazdelitev, na primer Poissonovo, gama ali normalno porazdelitev.

Za porazdelitve iz družine eksponentnih porazdelitev veljata naslednji lastnosti:

- Porazdelitev je enolično določena s parametroma θ_i in ϕ.
- Varianca naključne spremenljivke Y_i je funkcija odvisna od povprečne vrednosti $\mu_i = E(Y_i)$, kar lahko zapišemo kot

  $$Var(Y_i) = \frac{\phi v(\mu_i)}{\omega_i},$$

  kjer je v variančna funkcija, tj. gladka funkcija, ki varianco naključne količine prikazuje kot funkcijo svojega povprečja, ϕ je parameter disperzije in ω_i je utež za opazovanje i [5].

Utež nam sporoči pomembnost posamezne informacije. Posplošeni linearni model upošteva bolj tiste informacije, ki imajo večjo utež. Izbira uteži je odvisna od tega, kaj želimo modelirati. Pri modeliranju škodne pogostosti imamo na primer eno polico, ki traja samo en mesec in drugo polico, ki traja eno leto. Predstavljamo si lahko, da je za zavarovalnico pomembno, koliko časa je neka polica aktivna, saj je zavarovalnica takrat izpostavljena nekemu tveganju. Zato pri modeliranju škodne pogostosti za utež izberemo kar izpostavljenost. Pri modeliranju povprečne škode, kjer trajanje police ni pomembno, pa je pomembno,

koliko škod se je zgodilo na določeni polici. Torej lahko pri modeliranju povprečne škode za utež izberemo število škod.

Poglejmo si nekaj primerov variančnih funkcij, ki se v zavarovalništvu največkrat pojavijo [4]:

Porazdelitev	Variančna funkcija $v(x)$
Normalna	1
Poissonova	x
Gamma	x^2
Inverzna Gaussova	x^3

Tabela 3.1: Variančne funkcije.

Zraven variančne funkcije v in uteži ω_i se pri definiciji variance za posamezno opazovanje pojavi še ϕ. Včasih je $\phi = 1$ in zato izpade iz modela. V splošnem pa je ta parameter neznan, zato ga je potrebno iz danih podatkov oceniti.

3.2 Sistematična komponenta

Linearni prediktor η je definiran kot linearna kombinacija pojasnjevalnih spremenljivk $X_1, \ldots, X_n$:

$$\eta = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n.$$

Ta komponenta vključi informacije o pojasnjevalnih spremenljivkah v naš model. Linearni prediktor η je preko povezovalne funkcije g povezan s pričakovano vrednostjo odvisne spremenljivke Y.

3.3 Povezovalna funkcija

Povezovalna funkcija g je odvedljiva in monotona funkcija, ki povezuje naključno in sistematično komponento na naslednji način:

$$g(\mu) = \eta,$$

kjer je $\mu = E(Y)$. V praksi pa je bolj uporabno gledati na μ kot na funkcijo linearnega prediktorja η:

$$E(Y) = \mu = g^{-1}(\eta).$$

Kar pomeni, da je pričakovana vrednost odvisne spremenljivke lahko neka transformacija linearne kombinacije pojasnjevalnih spremenljivk.

V teoriji lahko za vsako opazovanje uporabimo drugo povezovalno funkcijo, vendar se v praksi to ne dela. V tabeli 3.2 imamo prikazanih nekaj značilnih povezovalnih funkcij, ki se uporabljajo v zavarovalništvu in tudi drugje [4]:

	$g(x)$	$g^{-1}(x)$
Identiteta	x	x
Logaritemska	$\ln(x)$	e^x
Obratna	$1/x$	$1/x$
Logit	$\ln(x/(1-x))$	$e^x/(1+e^x)$

Tabela 3.2: Povezovalne funkcije.

V zavarovalništvu se mnogokrat uporablja logaritemska povezovalna funkcija, saj se z njeno uporabo varianca zmanjša ter pozitivne vrednosti ostanejo pozitivne. S to funkcijo lahko ustvarimo multiplikativni posplošeni linearni model, ki ga zapišemo na naslednji način [4]:

$$E(Y_i) = \mu_i = g^{-1}(\beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) = e^{\beta_1 x_{i1}} e^{\beta_2 x_{i2}} \cdots e^{\beta_p x_{ip}}.$$

3.4 Modeliranje višine škode s posplošenim linearni modelom

Za modeliranje višine škode z GLM-jem obstaja več modelov, ki se razlikujejo glede na izbrano porazdelitev za višino škode. Te porazdelitve so lahko Poissonova porazdelitev, gama porazdelitev in nekatere druge. Vemo, da mora porazdelitev višine škode zavzemati nenegativne vrednosti in da mora biti asimetrična v desno, saj so velike škode redke, medtem ko so relativno majhne škode zelo pogoste. Zato običajno predpostavimo, da je posamezna škoda porazdeljena po gama porazdelitvi, ki ima gosoto verjetnosti enako

$$p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \qquad x > 0,$$

kjer je Γ funkcija gama in $\alpha, \beta > 0$.

Vemo, da za gama porazdelitve velja naslednja trditev [5]:

Trditev 3.1 *Naj bodo $X_1, X_2, \ldots, X_n$ gama porazdeljene neodvisne naključne spremenljivke, kjer $X_1 \sim \Gamma(\alpha_1, \beta), X_2 \sim \Gamma(\alpha_2, \beta), \ldots, X_n \sim \Gamma(\alpha_n, \beta)$. Potem je vsota teh naključnih*

spremenljivk $S = X_1 + X_2 + \cdots + X_n$ *tudi gama porazdeljena in sicer:*

$$S \sim \Gamma(\alpha_1 + \alpha_2 + \cdots + \alpha_n, \beta).$$

Naj bo X vsota ω škod, ki so neodvisne $\Gamma(\alpha, \beta)$ porazdeljene naključne spremenljivke, potem po zgornji trditvi velja, da je $X \sim \Gamma(\omega\alpha, \beta)$. Zato lahko gosoto verjetnosti za naključno spremenljivko $Y = X/\omega$ zapišemo kot

$$p_Y(y) = \omega p_X(\omega y) = \frac{(\omega\beta)^{\omega\alpha}}{\Gamma(\omega\alpha)} y^{\omega\alpha-1} e^{-\omega\beta y}, \qquad y > 0.$$

Torej je $Y \sim \Gamma(\omega\alpha, \omega\beta)$ in pričakovana vrednost od Y je enaka α/β. Ker želimo zgornjo gostoto verjetnosti preoblikovati v obliko (3.1), napravimo reparametrizacijo: $\mu = \alpha/\beta$ in $\phi = 1/\alpha$. Ker sta $\alpha, \beta > 0$, sta tudi $\mu, \phi > 0$. Gostoto verjetnosti naključne spremenljivke Y lahko sedaj zapišemo kot

$$\begin{aligned} p_Y(y) = p_Y(y; \mu, \phi) &= \frac{1}{\Gamma(\omega/\phi)} \left(\frac{\omega}{\mu\phi}\right)^{\omega/\phi} y^{(\omega/\phi)-1} e^{-\omega y/(\mu\phi)} \\ &= \exp\left(\frac{-y/\mu - \log(\mu)}{\phi/\omega} + c(\phi, \omega; y)\right), \qquad y > 0, \end{aligned}$$

kjer je $c(\phi, \omega; y) = \frac{\omega \log(\omega y/\phi)}{\phi} - \log(y) - \log(\Gamma(\omega/\phi))$.
Z dodatno reparametrizacijo $\theta = -1/\mu$ dobimo $\theta < 0$ ter naslednjo gostoto verjetnosti naključne spremenljivke Y:

$$p_Y(y; \theta, \phi) = \exp\left(\frac{y\theta + \log(-\theta)}{\phi/\omega_i} + c(\phi, \omega; y)\right).$$

Opazimo, da gama porazdelitev sodi v družino eksponentno porazdeljenih naključnih spremenljivk, kjer je $b(\theta) = -\log(-\theta)$. Zato lahko gama porazdelitev uporabimo pri modeliranju višine škode s posplošenim linearnim modelom.

Poglavje 4

Strojno učenje

4.1 Uvod in motivacija

Velik razlog za razvoj strojnega učenja v zadnjih letih je ta, da smo trenutno v dobi podatkov. To pomeni, da količina podatkov v naših življenjih vseskozi narašča. Današnji računalniki, trdi diski in shrambe na spletu nam omogočajo shranjevanje tako pomembnih podatkov kot tudi podatkov, ki bi jih sicer zavrgli. Današnja elektronika, ki je skoraj povsod, nenehno shranjuje naše preference, odločitve, stvari, ki jih maramo in stvari, ki jih sovražimo. Zgoraj napisano je gledano le na nivoju posameznika, zato si količine podatkov, ki se shranjujejo na primer pri poslovanju, ne moremo niti predstavljati.

V teh podatkih se lahko seveda skrivajo uporabne informacije. Namen podatkovnega rudarjenja je ta, da iz ogromne količine podatkov razberemo, kaj je pomembno oziroma kateri podatki vsebujejo največ uporabnih informacij. To imenujemo prepoznavanje vzorcev, s čimer se je človek ukvarjal že od samega začetka svojega nastanka. Lovci so iskali vzorce pri obnašanju živali in kmetje so opazovali vzorce pri rasti pridelkov. Eden od glavnih korakov pri rudarjenju podatkov je strojno učenje. Strojno učenje se ukvarja s preučevanjem in razvojem algoritmov, ki se učijo iz podatkov, in uporabo teh algoritmov za napovedovanje prihodnosti.

Iskanje vzorcev v podatkih samo po sebi ni nekaj novega. Prognostiki, statistiki in ekonomisti že dolgo časa delajo na tem, da bi vzorce med podatki lahko prepoznali in jih nato uporabili pri napovedovanju. Ker se je v zadnjem času količina shranjenih podatkov zelo povečala, sta strojno učenje in rudarjenje podatkov prišla v ospredje. Zaradi ogromnega števila podatkov v podatkovnih skladiščih, je naša edina možnost za iskanje vzorcev podatkovno rudarjenje. Z dobro analizo podatkov lahko le-ti postanejo dragocen vir informacij,

kar pa ima lahko velik vpliv na napovedi, zlasti v poslovnih krogih. V nadaljevanju bomo definirali pojem podatkovnega rudarjenja.

Podatkovno rudarjenje je proces odkrivanja vzorcev v podatkih. Ta proces mora biti avtomatičen ali delno avtomatičen, najdeni vzorci pa morajo biti uporabni, oziroma nam morajo dati neko prednost, ki je običajno ekonomska [10].

Podatkovno rudarjenje je torej namenjeno reševanju problemov s pomočjo analize podatkov, ki se nahajajo v podatkovnih skladiščih. Nato lahko s pomočjo te analize napovemo nove situacije, ki se lahko zgodijo, oziroma napovemo lastnosti za nove podatke, ki prispejo v podatkovno skladišče.

Poglejmo si to na primeru problema zvestobe strank. Neko podatkovno skladišče z informacijami o posamezniku in njegovi uporabi storitev lahko poda odgovor na vprašanje, ali bo stranka zapustila ponudnika ali ne. Na osnovi preteklih podatkov lahko s pomočjo strojnega učenja in podatkovnega rudarjenja prepoznamo lastnosti, ki so značilne za zveste stranke. Odkrijemo pa lahko tudi lastnosti, ki so značilne za stranke, ki želijo ponudnika zamenjati. Ko te lastnosti najdemo, jih lahko uporabimo za identifikacijo zvestih strank ter rizičnih strank, ki želijo zamenjati ponudnika. Rizičnim strankam lahko nato ponudimo ugodnosti ter si tako vsaj za nekatere zagotovimo, da ne zamenjajo ponudnika. Te lastnosti lahko uporabimo tudi za privabitev novih strank.

Nekaj podobnega bomo poskušali izvesti v tej magistrski nalogi. Ukvarjali se bomo z avtomobilskim zavarovanjem, kjer bomo poskušali na podlagi lastnosti posameznika in njegovega vozila napovedati višino škode.

4.2 Strojno učenje

V življenju smo se že vsi morali kaj naučiti in vsak od nas je razvil svoj način učenja novih stvari. Računalnik pa za učenje uporablja drugačen pristop kot človek. Iz zgornje definicije o podatkovnem rudarjenju vemo, da se strojno učenje ukvarja s preučevanjem in razvojem algoritmov za odkrivanje vzorcev. Tako lahko strojno učenje opišemo na naslednji način [10]:

Stvari se naučijo, ko se njihovo obnašanje spremeni na tak način, da naloge v prihodnosti opravijo bolje.

Torej je učenje računalnika ali stroja bolj povezano z uspešnostjo, kot s pridobivanjem znanja. Znanje računalnika preverimo tako, da trenutno obnašanje primerjamo z obnašanjem v preteklosti.

Probleme strojnega učenja lahko razdelimo v tri skupine: nadzorovano učenje, nenadzorovano učenje in delno-nadzorovano učenje. Pri nadzorovanem učenju je vsako opazovanje oziroma primer $(x_1, x_2, \ldots, x_n)$ povezano z vrednostjo izhodne spremenljivke $(y_1, y_2, \ldots, y_n)$. Cilj je zgraditi model, ki bo opazovanja povezal z izhodno spremenljivko, z željo, da bodo napovedi izhodne spremenljivke za nove podatke pravilne. V kategorijo nadzorovanega učenja spadajo naslednje metode: logistična regresija, linearna regresija, metoda podpornih vektorjev in nevronske mreže. Nekatere modele si bomo podrobneje ogledali kasneje, zato si poglejmo sedaj še nenadzorovano učenje. Nenadzorovano učenje zajema drugačne probleme. Tudi tukaj imamo opazovanja $(x_1, \ldots, x_n)$, nimamo pa povezave z vrednostmi izhodne spremenljivke $(y_1, \ldots, y_n)$. Pravimo, da na teh podatkih delamo na slepo. Takoj se pojavi vprašanje, katero metodo strojnega učenja pa uporabimo v tem primeru. Ena možnost je iskanje povezave med vhodnimi spremenljivkami. Pogosta metoda pri nenadzorovanem učenju je metoda razvrščanja v skupine. Njen cilj je ugotoviti, ali lahko na podlagi opazovanj najdemo kakšne skupine, ki so si med seboj dovolj podobne oziroma so si med seboj dovolj različne. Večina problemov torej spada v nadzorovano učenje ali nenadzorovano učenje. Obstajajo pa problemi, ki ne spadajo v nobeno od zgornjih kategorij, zato obstaja še ena skupina, ki jo imenujemo delno-nadzorovano učenje in ta se pojavi, ko imamo vrednosti izhodne spremenljivke le za del opazovanj. Recimo, da imamo n opazovanj in za m teh opazovanj $(m < n)$ imamo vrednosti izhodne spremenljivke, za ostalih $(n - m)$ opazovanj pa nimamo vrednosti izhodne spremenljivke. Zato želimo v tem primeru uporabiti strojno učenje, ki bo hkrati upoštevalo m opazovanj, katera imajo vrednost izhodne spremenljivke, in tudi $(n - m)$ opazovanj, pri katerih nimamo vrednosti izhodne spremenljivke.

Poglejmo si sedaj nadzorovano strojno učenje z matematičnega vidika.

4.3 Nadzorovano strojno učenje

Iz matematičnega vidika je strojno učenje na začetku dokaj enostavno. Prvo moramo imeti lepo urejene podatke. To ne pomeni, da so podatki urejeni po velikosti, ampak, da so porazdeljeni po matriki podatkov, v kateri vrstice predstavljajo posamezna opazovanja oziroma primere, stolpci pa lastnosti oziroma spremenljivke. Če se spomnimo primera problema zvestobe strank, bi ena vrstica matrike predstavljala enega posameznika, en stolpec pa eno njegovo lastnost. Naši podatki torej izgledajo tako [6]:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}.$$

Ena vrstica predstavlja en primer, kar pomeni, da imamo n primerov. En stolpec predstavlja eno lastnost oziroma spremenljivko, to pomeni, da imamo na voljo p spremenljivk. Oznaka za vrstico je x_i, $i = 1, 2, \ldots, n$, za stolpec pa X_j, $j = 1, 2, \ldots, p$. Običajno velja, da je $n > p$, saj lahko opazovanja hitro pridobimo. Vendar to ne pomeni, da je število p nujno majhno. Na primer v medicini je lahko število spremenljivk ogromno, govorimo o stotisočih ali celo milijonih. Spremenljivkam X_j pravimo tudi vhodne spremenljivke. Zgornjo matriko imenujemo vhodni podatki in zapišemo jo lahko kot

$$X = (X_1, X_2, \ldots, X_p).$$

Pri nadzorovanem učenju moramo seveda imeti tudi izhodno ali ciljno spremenljivko, ki jo želimo napovedati na osnovi vhodnih spremenljivk. Vhodni podatki temeljijo na preteklosti, zato za vsako opazovanje poznamo zraven lastnosti tudi vrednost izhodne spremenljivke. Zato se matrika podatkov največkrat zapiše v naslednji obliki [6]:

$$\left[\begin{array}{cccc|c} x_{1,1} & x_{1,1} & \cdots & x_{1,p} & y_1 \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} & y_n \end{array}\right].$$

Cilj strojnega učenje je poiskati funkcijo f, ki predstavlja povezavo med vhodnimi podatki in izhodno spremenljivko. To lahko zapišemo kot

$$Y = f(X) + \epsilon = f(X_1, \ldots, X_p) + \epsilon,$$

kjer je f neznana funkcija odvisna od $X_1, X_2, \ldots, X_p$, in ϵ naključna napaka s povprečjem 0, ki je neodvisna od X [6]. Funkcija f predstavlja sistematično informacijo, ki jo X predlaga o Y, oziroma funkcija f povezuje vhodne podatke z izhodno spremenljivko.

Naloga strojnega učenja je oceniti funkcijo f. Izbira načina ocenjevanja funkcije f porodi različne metode strojnega učenje, zato si najprej poglejmo, zakaj funkcijo f ocenjujemo.

4.3.1 Namen ocenjevanja funkcije f

Funkcijo f ocenjujemo za napovedovanje in sklepanje. Oboje si poglejmo bolj podrobno.

Napovedovanje

Vhodne podatke X lahko običajno zlahka pridobimo, problem pa nastane s pridobivanjem izhodnih podatkov Y, saj le-teh včasih preprosto ne moremo dobiti. V tem primeru, ker je povprečje naključnih napak enako 0, lahko napovemo Y z uporabo

$$\hat{Y} = \hat{f}(X),$$

kjer $\hat{f}$ predstavlja oceno za funkcijo f in $\hat{Y}$ pripadajoče napovedi za Y [6]. V tem primeru obravnavamo $\hat{f}$ kot črno škatlo, kar pomeni, da nas ne zanima oblika funkcije $\hat{f}$, ampak kako dobro ta funkcija napoveduje Y.

Točnost $\hat{Y}$, kot napoved za Y, je odvisna od reducibilne in nereducibilne napake. Funkcija $\hat{f}$ nikoli ne bo popolna ocena za funkcijo f, zato pride do napake. To napako lahko reduciramo oziroma zmanjšamo z uporabo ustrezne učne tehnike in posledično izboljšamo točnost funkcije $\hat{f}$. Vendar tudi če najdemo funkcijo g, ki bi perfektno ocenila f, torej da bi za napoved $\hat{Y} = g(X)$ veljalo $\hat{Y} = f(X)$, bi ta napoved še vedno vsebovala napako, saj je Y odvisna tudi od ϵ, ki pa ga po definiciji ne moremo napovedati na osnovi X. To napako imenujemo nereducibilna napaka, ker je ne moremo odpraviti. Zlahka lahko pokažemo naslednje [6]:

$$E[(Y - \hat{Y})^2] = E[(f(X) + \epsilon - \hat{f}(X))^2] = (f(X) - \hat{f}(X))^2 + Var(\epsilon),$$

kjer je $E[(Y - \hat{Y})^2]$ pričakovana vrednost kvadrata razlike med dejansko vrednostjo Y in napovedano vrednostjo $\hat{Y}$, $(f(X) - \hat{f}(X))^2$ reducibilna napaka ter $Var(\epsilon)$ nereducibilna napaka.

Sklepanje

Zraven napovedovanja nas velikokrat zanima tudi, kako sprememba naših vhodnih podatkov X vpliva na Y. V tem primeru želimo oceniti f, ampak naš cilj ni nujno napovedati Y, ampak ugotoviti in razumeti povezavo med vhodnimi podatki X in izhodno spremenljivko Y. Zdaj $\hat{f}$ ne moremo obravnavati kot črno škatlo, saj želimo vedeti njeno točno obliko. Tukaj se pojavijo naslednja vprašanja:

- Katere vhodne spremenljivke so povezane z izhodno spremenljivko?
  V večini primerov je le majhen del vhodnih spremenljivk občutno povezanih z Y. Iskanje teh najboljših spremenljivk se velikokrat izkaže za zelo uporabno.

- Kakšna je povezava med izhodno in posamezno vhodno spremenljivko?
  Nekatere vhodne spremenljivke imajo lahko pozitiven vpliv na izhodno spremenljivko, kar pomeni, da se s povečanjem vhodne spremenljivke poveča tudi izhodna spremenljivka, spet druge pa negativen vpliv. Glede na kompleksnost f je lahko povezava med izhodno in posamezno vhodno spremenljivko odvisna tudi od nekaterih drugih spremenljivk.

- Ali lahko povezavo med izhodno in posamezno vhodno spremenljivko ustrezno povzamemo z linearno odvisnostjo, ali pa je povezava bolj kompleksna?
  Zgodovinsko gledano večina metod za ocenjevanje f temelji na linearni odvisnosti. Sicer je to v nekaterih primerih ustrezno, vendar se pri večini izkaže, da linearna odvisnost ni ustrezna. Zato nam linearni model ne more dati dovolj dobre povezave, kar pomeni, da moramo poiskati bolj kompleksne povezave.

Torej, glede na to, ali je naš namen napovedovanje, sklepanje ali morda oboje, poznamo različne metode za ocenjevanje f. Kompleksne nelinearne metode imajo lahko odlične napovedi, vendar pa jih je v večini primerov zelo težko razumeti. Medtem pa nam linearni modeli omogočajo preprosta in razumljiva sklepanja, vendar pa so lahko njihove napovedi slabe.

4.3.2 Regresija in razvrščanje

Poznamo dve vrsti spremenljivk: kvalitativne (ali kategorične) in kvantitativne (ali numerične). Kvantitativne spremenljivke so na primer starost, višina, cena delnic itd. Kvalitativne spremenljivke pa zavzamejo neko vrednost, ki ni nujno število, na primer spol, znamka, vrsta zavarovanja itd.

Problem napovedovanja kvantitativne spremenljivke imenujemo problem regresije, medtem ko problem napovedovanja kvalitativne spremenljivke imenujemo problem razvrščanja. Pomembno je prepoznati, kateri problem obravnavamo, saj so nekatere metode (linearna regresija) namenjene samo regresijskim problemom, druge (logistična regresija) pa samo razvrščanju. Nekatere metode pa so uporabne za oba problema (nevronske mreže, odločitvena drevesa, metoda najbližjih sosedov, naključni gozdovi).

4.3.3 Ocenjevanje natančnosti modela

Da ocenimo učinkovitost napovednega modela, moramo izmeriti njegovo točnost oziroma natančnost. Za to potrebujemo način, kako primerjati napovedi modela z resničnimi podatki. Pri regresijskem problemu običajno uporabimo povprečno kvadratno napako, ki je

podana z naslednjim predpisom [6]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

kjer je y_i dejanska vrednost in $\hat{f}(x_i)$ napoved za i-to opazovanje. MSE bo majhen, če bodo napovedi in dejanske vrednosti zelo blizu, ter velik, če se bodo napovedi zelo razlikovale od dejanskih vrednosti. Najboljši regresijski model je tisti, ki ima najmanjšo vrednost MSE.

Pri problemu razvrščanja pa ne napovedujemo številke, ampak nek razred, zato ne moremo uporabiti MSE. Tukaj si pomagamo s stopnjo napake. Recimo, da želimo oceniti f na osnovi podatkov $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, kjer so y_i kvalitativne vrednosti. Stopnja napake je potem enaka [6]

$$err = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i),$$

kjer je y_i dejanski razred, $\hat{y}_i$ pa napovedani razred, pri uporabi funkcije $\hat{f}$, za i-to opazovanje. $I(y_i \neq \hat{y}_i)$ je indikatorska spremenljivka z naslednjim predpisom:

$$I(y_i \neq \hat{y}_i) = \begin{cases} 1 & \text{, če } y_i \neq \hat{y}_i \\ 0 & \text{, sicer} \end{cases}.$$

Najboljši model za razvrščanje je tisti, ki ima najmanjšo vrednost err.

Omenimo še, da poznamo dve vrsti napak, in sicer učno napako in testno napako. Imeni izvirata iz računanja napake na določeni podmnožici podatkov. V praksi imamo podatke razdeljene na učne in testne podatke. Na učnih podatkih z izbrano metodo ocenimo funkcijo f, ki jo potem na testnih podatkih preverimo. Pomembno je, da model v procesu učenja testnih podatkov ne vidi. Če izračunamo napako na učnih podatkih, potem tej napaki rečemo učna napaka in ta nas v resnici ne zanima. Nas bolj zanima napaka, ki jo izračunamo na testnih podatkih. To napako imenujemo testna napaka in za njen izračun se uporablja enaka formula kot za učno napako. Najboljši model je potem tisti, ki ima najmanjšo testno napako.

V praksi je izračun učne napake dokaj preprost. Računanje testne napake pa je velikokrat nemogoče, saj nimamo testne množice. Zato se hitro porodi ideja, da podatke, ki jih imamo na voljo, razdelimo na učne in testne podatke. Ta ideja je sicer dobra, vendar je kot metoda za preverjanje natančnosti modela dokaj šibka. V praksi se pogosto izkaže, da je pri enkratnem deljenju podatkov na dve množici testna napaka pristransko ocenjena, saj ima veliko varianco. Zato sta se za ocenjevanje testne napake uveljavili dve metodi

vzorčenja: k-kratno prečno preverjanje in zankanje. Obe metodi bomo podrobneje spoznali v nadaljevanju.

4.4 Metode vzorčenja

V moderni statistiki so metode vzorčenja pomembna orodja, ki se ukvarjajo s ponavljajočim vzorčenjem podatkov in modeliranjem na njih. S tem lahko pridobimo dodatne informacije o modelu. Na primer, da želimo oceniti variabilnost linearne regresije. Iz učnih podatkov lahko izbiramo različne vzorce, nato na vsakem vzorcu posebej izvedemo linearno regresijo ter na koncu preverimo razlike med dobljenimi modeli. Tako lahko dobimo dodatne informacije o modelu, ki jih ob uporabi vseh podatkov ne bi.

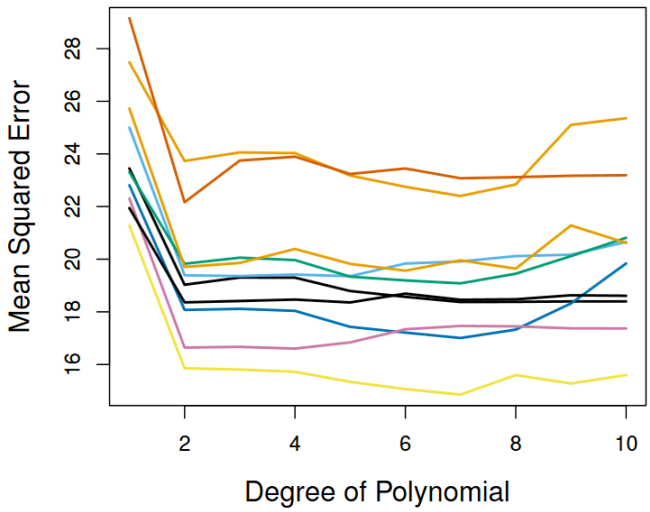
Metode vzorčenja so lahko računsko zahtevne, saj temeljijo na večkratni uporabi metode strojnega učenja na različnih podmnožicah učnih podatkov. Zaradi napredka pri hitrosti računalniškega računanja, pa so se v zadnjem času metode vzorčenja zelo razvile. V tem delu magistrske naloge bomo opisali dve najbolj pogosti metodi vzorčenja, to sta prečno preverjanje in zankanje. Obe metodi se uporabljata za ocenjevanje testne napake, z namenom primerjanja različnih modelov med seboj in izbire najboljšega modela.

4.4.1 Prečno preverjanje

Poznamo več metod prečnega preverjanja, vendar vse temeljijo na istem principu, tj. delitvi podatkov na več množic. Prečno preverjanje razdeli vhodne podatke na podmnožice in model nauči na delu teh podmnožic ter nato testno napako izračuna na preostalih podatkih, ki jih model ni uporabil za učenje. Metode prečnega preverjanja se razlikujejo glede na to, kako naredijo podmnožice in katere podmnožice uporabijo za računanje testne napake. V nadaljevanju si bomo pogledali tri metode: pristop s preveritveno množico, metoda "izpusti enega" ter k-kratno prečno preverjanje.

Pristop s preveritveno množico

S tem pristopu pravzaprav ne vzorčimo podatkov, ampak jih le razdelimo na dve množici. Ta pristop je zelo preprost, saj metoda naključno razdeli podatke na učno in preveritveno množico. Model naučimo na podatkih iz učne množice, testno napako pa izračunamo na preveritveni množici.



Slika 4.1: Pristop s preveritveno množico je bil izveden 10-krat. Vsakič je bila uporabljena druga naključna delitev podatkov na dve množici.

Na sliki 4.1 lahko opazimo, da je variabilnost testne napake med različnimi delitvami velika. Implementacija tega pristopa ni zahtevna, ima pa dve pomanjkljivosti:

- Iz slike 4.1 vidimo, da je izračun testne napake zelo odvisen od delitve podatkov na učno in preveritveno množico. Različne delitve lahko testno napako izboljšajo ali pa poslabšajo.
- Za učenje modela uporabimo le del podatkov, ki so nam na voljo. Ker pri strojnem učenju velja, da so modeli boljši, če imajo na voljo več podatkov za učenje, se lahko zgodi, da precenimo testno napako za model, ki je naučen na celotni množici podatkov.

V nadaljevanju bomo predstavili še dve metodi prečnega preverjanjanja, ki sta izboljšani verziji tega pristopa in se spopadeta z zgornjima pomankljivostima.

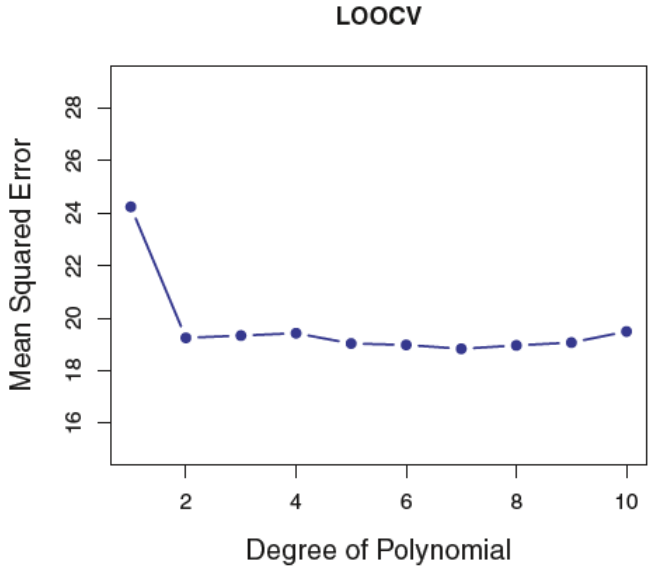
Metoda "izpusti enega"

Ta metoda je zelo podobna pristopu s preveritveno množico, vendar pa poskuša odpraviti njene pomanjkljivosti, ki smo jih omenili zgoraj. Recimo, da imamo na voljo podatke $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Tako kot pri prejšnjem pristopu, tudi s to metodo razdelimo podatke na dve množici. Ampak namesto da naredimo dve podobno veliki množici, za preveritveno množico vzamemo le eno opazovanje (x_1, y_1), medtem ko ostale podatke $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ uporabimo za učenje. Torej ta metoda uporabi za učenje

vsa opazovanja, razen enega, ter naredi napoved za izpuščeno opazovanje. Ker je opazovanje (x_1, y_1) izpuščeno iz učenja, je ocena testne napake pri regresiji enaka $MSE_1 = (y_1 - \hat{y_1})^2$. Čeprav je ocena testne napake nepristranska, je ta še vedno slaba, saj temelji le na enem opazovanju ter je zato močno variabilna. Postopek lahko ponovimo, vendar tokrat izberemo opazovanje (x_2, y_2) za preveritveno množico in model naučimo na množici $\{(x_1, y_1), (x_3, y_3), \ldots, (x_n, y_n)\}$ ter izračunamo $MSE_2 = (y_2 - \hat{y_2})^2$. Če to ponovimo n-krat, dobimo n kvadratov napak $MSE_1, \ldots, MSE_n$. Metoda "izpusti enega" izračuna končno testno napako kot povprečje teh testnih napak [6]:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i.$$

Tako odpravimo nekatere pomanjkljivosti pristopa s preveritveno množico. Ta metoda je bolj nepristranska kot prejšnja, saj prejšnja za učenje uporabi približno polovico podatkov, ki so na voljo. Tukaj pa za učenje modela uporabimo vse podatke razen enega, kar pomeni, da metoda "izpusti enega" ne preceni testne napake tako zelo, kot jo pristop s preveritveno množico. Naslednja prednost te metode je nenakljčnost, saj ponovna uporaba te metode na istih podatkih poda isti rezultat, kar pri prejšnji metodi ni veljalo, saj smo pri večkratni uporabi metode dobili zelo različne ocene testne napake. Grafični prikaz uporabe metode "izpusti enega" je prikazan na sliki 4.2.



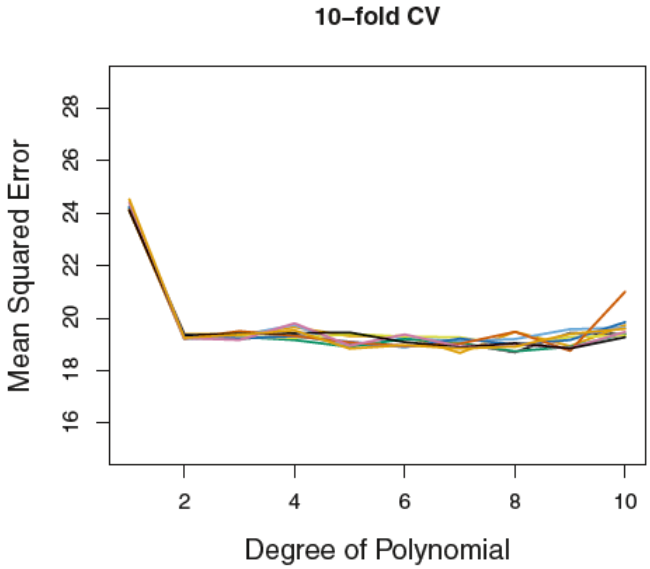
Slika 4.2: Metoda "izpusti enega".

k-kratno prečno preverjanje

Ta metoda je alternativa metodi "izpusti enega". Tukaj podatke $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ razdelimo v k množic, ki so približno enako velike. Prvo množico uporabimo za preveritveno množico, ostale množice pa uporabimo za učenje modela. Povprečno kvadratno napako MSE_1 nato izračunamo na prvi preveritveni množici. Proces ponovimo k-krat in pri tem vsakič za preveritveno množico izberemo drugo množico. Od tod tudi ime k-kratno prečno preverjanje. Tako dobimo k ocen testne napake $MSE_1, \ldots, MSE_k$. Metoda k-kratno prečno preverjanje izračuna končno testno napako kot povprečje teh vrednosti [6]:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i.$$

Opazimo, da je metoda "izpusti enega" posebni primer k-kratnega prečnega preverjanja, kjer je $k = n$. V praksi največkrat uporabljamo 10-kratno ali 5-kratno prečno preverjanje.



Slika 4.3: 10-kratno prečno preverjanje je bilo izvedeno 9-krat. Vsakič je bila uporabljena druga naključna delitev podatkov na 10 množic.

Prednost 10-kratnega prečnega preverjanja v primerjavi z metodo "izpusti enega" je manjša računska zahtevnost, saj mora metoda "izpusti enega" strojno učenje uporabiti n-krat, 10-kratno prečno preverjanje pa jo uporabi le 10-krat. V primeru ogromne količine podatkov, se ta prednost zelo pozna.

Na sliki 4.3 lahko opazimo, da je pri 10-kratnem prečnem preverjanju še vedno prisotno nekaj variabilnosti, saj podatke naključno delimo, vendar pa je ta močno zmanjšana v

primerjavi s pristopom s preveritveno množico.

Zgornje enačbe in primeri se nanašajo na probleme regresije. Seveda pa lahko zgornje metode uporabimo tudi na problemih razvrščanja. Spremenijo se le enačbe in v primeru metode "izpusti enega" dobimo

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i).$$

V primeru pristopa s preveritveno množico in k-kratnega prečnega preverjanja enačbe definiramo analogno.

4.4.2 Zankanje

Tudi tukaj vzorčimo podatke iz obstoječe množice podatkov. Glavna ideja zankanja je vzorčenje podatkov s ponavljanjem. Pri prejšnjih metodah smo podatke vzorčili brez ponavljanja – ko smo enkrat izbrali opazovanje za učno množico, tega opazovanja nismo smeli izbrati še enkrat. Pri zankanju pa lahko posamezno opazovanje izberemo tudi večkrat, lahko pa se zgodi, da nekatero opazovanje sploh ne izberemo. Neizbrana opazovanja lahko zato uporabimo za preveritveno množico.

Tukaj se pojavi vprašanje, kolikšna je verjetnost, da posamezno opazovanje ne bo izbrano v učno množico, ki vsebuje n opazovanj, če imamo na voljo n opazovanj. Vemo, da je verjetnost izbire enega opazovanja enaka $1/n$, kar pomeni, da je verjetnost neizbire enega opazovanja enaka $(1 - 1/n)$. Če je n velik, dobimo naslednje [10]:

$$\left(1 - \frac{1}{n}\right)^n \simeq e^{-1} \simeq 0.368.$$

To predstavlja verjetnost, da posamezno opazovanje ne bo izbrano v učno množico velikosti n, kar pomeni, da bo v primeru velikega števila opazovanj učna množica vsebovala približno 63.2% vseh opazovanj, preveritvena množica pa 36.8%.

Če sedaj model učimo na dobljeni učni množici in testno napako izračunamo na dobljeni preveritveni množici, bo ta napaka pesimistična ocena za testno napako, saj ima učna množica, kljub temu da vsebuje n opazovanj, le 63.2% vseh opazovanj. Za primerjavo je pri 10-kratnem prečnem preverjanju ta odstotek enak 90%. Torej moramo izračunano napako nekoliko popraviti. Vemo, da je učna napaka zelo optimistična ter je kot samostojno merilo ne smemo uporabiti za določanje natančnosti modela. Zato jo pri zankanju združimo s

testno napako in tako popravimo izračunano testno napako na naslednji način [10]:

$$Err_{Z_i} = 0.632 \cdot Err_{\text{testna napaka}} + 0.368 \cdot Err_{\text{učna napaka}},$$

kjer Err_{Z_i} predstavlja popravljeno testno napako pri delitvi Z_i. Ta popravek temelji na tem, da je verjetnost odsotnosti opazovanja iz učne množice enaka 0.368.

Z različnim vzorčenjem ta postopek ponovimo B-krat in končno testno napako pri zankanju dobimo kot povprečje teh napak:

$$Err = \frac{1}{B} \sum_{i=1}^{B} Err_{Z_i}.$$

Poglavje 5

Odločitvena drevesa

Do tega poglavja smo predstavili, kaj pomeni strojno učenje in kako izbrati najboljši model. V nadaljevanju pa se bomo osredotočili na to, kako zgradimo različne modele in kateri modeli so na voljo.

Za začetek si bomo podrobneje pogledali odločitvena drevesa, ki jih lahko uporabimo tako za probleme razvrščanja kot tudi za probleme regresije.

5.1 Regresijska drevesa

Za lažje razumevanje odločitvenih dreves si poglejmo naslednji primer, ki je izvzet iz [6]. V tabeli 5.1 imamo podatke o baseball igralcih, s pomočjo katerih želimo napovedati plačo igralca. Na voljo sta vhodni spremenljivki *Leta*, ki predstavlja število let igranja ter *Udarci*, ki predstavlja število udarcev v pretekli sezoni.

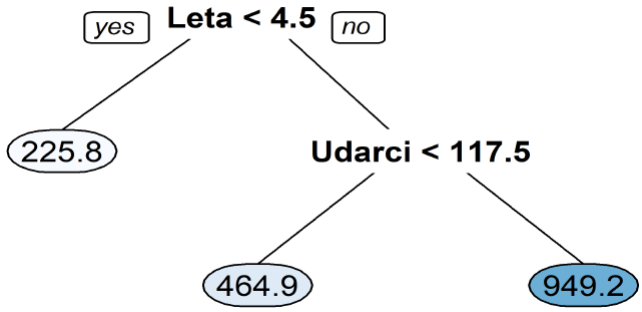
Igralec	**Plača** (v tisoč dolarjih)	**Udarci**	**Leta**
Alan Ashby	475.00	81	14
Alvin Davis	480.00	130	3
Andre Dawson	500.00	141	11
Andres Galarraga	91.50	87	2
Alfredo Griffin	750.00	169	11
⋮	⋮	⋮	⋮

Tabela 5.1: Podatki za primer.

Ustvariti želimo napovedni model za višino plače novega igralca. Na sliki 5.1 je prikazan primer regresijskega drevesa, ki ga lahko sestavimo iz naših podatkov. Drevo je sestavljeno

iz delitvenih pravil, ki se začnejo na vrhu drevesa. Prva delitev nam dodeli opazovanja, ki imajo spremenljivko $Leta < 4.5$, na levo vejo in napovedana plača za te igralce je enaka povprečju plač vseh igralcev, ki imajo $Leta < 4.5$. Tem igralcem bi torej ponudili plačo v višini 225.80 tisoč dolarjev. Vsa opazovanja, ki imajo $Leta > 4.5$, pa dodelimo na desno vejo drevesa. Ta veja se naprej deli glede na spremenljivko *Udarci*. Tako nam to drevo igralce razdeli v tri skupine:

- igralce, ki igrajo 4 leta ali manj,
- igralce, ki igrajo 5 let ali več in so v pretekli sezoni imeli manj kot 118 udarcev,
- igralce, ki igrajo 5 let ali več in so v pretekli sezoni imeli 118 udarcev ali več.



Slika 5.1: Regresijsko drevo.

To pomeni, da smo napovedni prostor razdelili v tri skupine, kar lahko zapišemo kot

- $R_1 = \{X \mid Leta < 4.5\}$,
- $R_2 = \{X \mid Leta \geq 4.5, Udarci < 117.5\}$,
- $R_3 = \{X \mid Leta \geq 4.5, Udarci \geq 117.5\}$.

Pripadajoče napovedane plače za te razrede so: 225.80, 464.90 in 949.20 tisoč dolarjev.

Skupine R_1,R_2 in R_3 imenujemo listi ali končna vozlišča drevesa. Običajno se drevesa rišejo obrnjena na glavo, tako da so listi na dnu. Točke v drevesu, kjer se napovedni prostor razdeli, pa imenujemo notranja vozlišča. V našem primeru imamo dva notranja vozlišča, ki vsebujeta besedila $Leta < 4.5$ in $Udarci < 117.5$. Povezavi med vozliščema pravimo veja.

Za razliko od posplošenih linearnih modelov, lahko zgrajeno odločitveno drevo preprosto razložimo. V našem primeru opazimo, da je za določanje višine plače najpomembnejša

spremenljivka *Leta*, saj imajo igralci z manj izkušnjami običajno nižje plače kot igralci, ki imajo več izkušenj. Vidimo tudi, da za igralce z manj izkušnjami, število udarcev nima pomembne vloge pri določanju višine njihovih plač. Pri izkušenih igralcih pa število udarcev v prejšnji sezoni zelo vpliva na višino njihove plače, saj več doseženih udarcev zvišuje plačo izkušenim igralcem. Je pa ta primer enostaven, zato je njegovo interpretacijo preprosto zapisati. Poglejmo si sedaj, kako takšno drevo zgradimo.

5.1.1 Gradnja drevesa za problem regresije

Pri gradnji drevesa imamo dva koraka [6]:

1. Razdelimo napovedni prostor, tj. množica možnih vrednosti za vhodne spremenljivke $X_1, X_2, \ldots, X_p$, na J disjunktnih skupin $R_1, \ldots, R_J$.
2. Za vsako opazovanje, ki je v skupini R_i, naredimo enako napoved. Ta napoved je povprečna vrednost izhodne spremenljivke vseh opazovanj v učni množici, ki so v skupini R_i.

Pojavi se vprašanje, kako skonstruirati skupine $R_1, R_2, \ldots, R_J$. V teoriji imajo te skupine lahko poljubno obliko, vendar si zaradi preprostosti in lažje interpretacije drevesa izberemo, da bodo imele skupine obliko večdimenzionalnih pravokotnikov ali škatel. Cilj je poiskati škatle $R_1, \ldots, R_J$, ki minimizirajo RSS, ki je podan kot [6]:

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

kjer je $\hat{y}_{R_j}$ povprečje izhodne spremenljivke izračunano na učnih podatkih v j-ti škatli, y_i pa i-to opazovanje.

Žal je nepredstavljivo, da obravnavamo vse možne razdelitve napovednega prostora v J škatel, zato naredimo delitev v skupine s pomočjo požrešnega pristopa, ki ga imenujemo rekurzivno binarno deljenje. Ime izhaja iz tega, da na vsakem koraku gradnje drevesa izberemo najboljšo delitev na trenutnem koraku, ki nam razdeli podatke v dve skupini. Torej nas ne zanima najboljša delitev, ki bi kasneje morda privedla do boljšega drevesa, ampak najboljša delitev na trenutnem koraku.

Da lahko izvedemo rekurzivno binarno deljenje, moramo izbrati spremenljivko X_j in število s tako, da bo delitev napovednega prostora na skupini $\{X \mid X_j < s\}$ in $\{X \mid X_j \geq s\}$ privedla do največjega možnega znižanja vrednosti RSS. To storimo tako, da naredimo binarne delitve za vse spremenljivke $X_1, \ldots, X_p$ in vse možne vrednosti s ter nato izberemo

tisto delitev, pri kateri je vrednost RSS najmanjša. Torej za vsak j in s definiramo par polovic

$$R_1(j,s) = \{X \mid X_j < s\},$$

$$R_2(j,s) = \{X \mid X_j \geq s\}$$

in iščemo takšni vrednosti j in s, ki minimizirata izraz

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

kjer je $\hat{y}_{R_1}$ povprečje izhodne spremenljivke izračunano na učnih podatkih v skupini $R_1(j,s)$ in $\hat{y}_{R_2}$ povprečje izhodne spremenljivke izračunano na učnih podatkih v skupini $R_2(j,s)$ [6]. Vrednosti j in s, ki minimizirata zgornji izraz, lahko hitro izračunamo, če število spremenljivk p ni preveliko.

Po enkratni izvedbi zgornjega procesa dobimo dve skupini. Ta proces lahko ponovimo na vsaki podskupini posebej ter tako dobimo najboljšo delitev za vsako podskupino. To ponavljamo, dokler ne pridemo do ustavitvenega pogoja, ki si ga izberemo sami. Lahko je na primer to trenutek, ko nobena od skupin ne vsebuje več kot 10 opazovanj.

Ko imamo narejene skupine $R_1, \ldots, R_J$, izračunamo povprečno vrednost izhodne spremenljivke za učne podatke v posamezni skupini in to vrednost določimo za našo napoved za posamezno skupino.

5.1.2 Obrezovanje dreves

S postopkom, ki smo ga opisali v podpoglavju 5.1.1, lahko pridemo do dobrih rezultatov na učnih podatkih, vendar se lahko zgodi, da bodo napovedi za testne podatke slabe, saj se bo model preveč prilagajal učnim podatkom. Vzrok za to je lahko preveč kompleksno drevo. Manjše število skupin $R_1, \ldots, R_J$ oziroma manjše drevo lahko privede do manjše variance in boljše interpretacije drevesa.

Možna rešitev, kako to odpraviti, je gradnja drevesa, dokler je zmanjšanje vrednosti RSS zaradi nove delitve dovolj veliko. To nas bo pripeljalo do manjšega drevesa, vendar to drevo ne bo nujno dobro. Zgodi se lahko, da že na začetku izberemo slabo delitev, ki bi ji sledila dobra delitev in zato bi se RSS zelo zmanjšal. Ampak tako dobljeno drevo ne bi dalo najboljših rezultatov. Boljša strategija je, da zgradimo celotno drevo T_0, kot je opisano v podpoglavju 5.1.1 in nato to drevo obrežemo, da dobimo poddrevo.

V nadaljevanju želimo določiti najboljši način obrezovanja drevesa. Intuitivno, naš cilj je poiskati poddrevo z najmanjšo testno napako. Če imamo podano poddrevo, potem lahko

ocenimo njegovo testno napako s prečnim preverjanjem. Vendar je v primeru velikega števila poddreves izvedba prečnega preverjanja za vsako poddrevo nepraktična. Poiskati moramo način, kako izbrati le nekaj možnih poddreves, na katerih bomo ocenili testno napako. Možna rešitev je metoda obrezovanja na osnovi cene zahtevnosti, znana tudi kot metoda obrezovanja na osnovi najšibkejše povezave. Namesto da gledamo vsa možna poddrevesa, nas zanima samo zaporedje poddreves, ki so indeksirani z nenegativnim (nastavitvenim) parametrom α. Vsaka vrednost parametra α ustreza določenemu poddrevesu $T \subset T_0$, ki minimizira naslednjo vrednost [6]:

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

Pri tem $|T|$ predstavlja število listov poddrevesa T, R_m je škatla, ki pripada m-temu listu ter $\hat{y}_{R_m}$ je napoved izhodne spremenljivke za škatlo R_m.

Parameter α nadzoruje kompromis med prilagoditvijo poddrevesa učnim podatkom in kompleksnostjo poddrevesa. Ko je $\alpha = 0$, potem je poddrevo T enako prvotnemu drevesu T_0, saj zgornji izraz predstavlja učno napako. Z naraščanjem α pa dodajamo kazen drevesom z velikim številom listov, zato bo zgornji izraz načeloma manjši za manjša poddrevesa. Izkaže se, da z naraščanjem α obrezovanje vej poteka po predvidljivem vrstnem redu, zato lahko enostavno pridobimo zaporedje poddreves kot funkcijo odvisno od α.

Postopek gradnje regresijskega drevesa z obrezovanjem lahko opišemo z naslednjim algoritmom [6]:

1. Uporabimo rekurzivno binarno deljenje za gradnjo velikega drevesa na učnih podatkih. Ustavimo se, ko vsak list vsebuje manj kot neko določeno število opazovanj.

2. Na zgrajenem drevesu uporabimo metodo obrezovanja na osnovi cene zahtevnosti, da dobimo zaporedje najboljših poddreves kot funkcijo parametra α.

3. Za izbiro parametra α uporabimo k-kratno prečno preverjanje. To pomeni, da razdelimo učne podatke na k množic in za vsak $i = 1, \ldots, k$:

   - Ponovimo koraka 1 in 2 na vseh množicah razen na i-ti.
   - Izračunamo MSE za podatke v izpuščeni i-ti množici, kot funkcijo odvisno od α.

   Izračunamo povrečje rezultatov za vsako vrednost α in izberemo α tako, da minimiziramo povprečno napako.

4. Vrnemo poddrevo, ki pripada izbrani vrednosti α, iz koraka 2.

5.2 Klasifikacijska drevesa

Odločitvena drevesa lahko uporabimo tudi za probleme razvrščanja. V tem primeru je spremenljivka, ki jo napovedujemo, kategorična in ne numerična, kot je v primeru regresije. Razlika je tudi v načinu napovedovanja, saj pri kategorični spremenljivki ne moremo izračunati njene povprečne vrednosti. Tukaj uporabimo pravilo največjega števila glasov, kar pomeni, da za napoved v posameznem listu določimo razred, tj. vrednost izhodne spremenljivke, ki je v listu najbolj zastopan.

Gradnja klasifikacijskih dreves poteka podobno kot gradnja regresijskih dreves. Tudi tukaj si pomagamo z rekurzivnim binarnim deljenjem, vendar pa za kriterij deljenja ne moremo uporabiti vrednosti RSS. Alternativa vrednosti RSS je napaka razvrščanja. Ker bomo napovedovali s pomočjo pravila največjega števila glasov, bomo pri napaki razvrščanja upoštevali delež napačno razvrščenih učnih podatkov v posameznem listu [6]:

$$E_m = 1 - \max_k (\hat{p}_{mk}),$$

kjer $\hat{p}_{mk}$ predstavlja delež učnih podatkov v m-tem listu, ki pripadajo k-temu razredu. Izkaže se, da napaka razvrščanja ni dovolj občutljiva za gradnjo dreves, zato sta se v praksi uveljavili drugi dve meri, to sta Gini-jev indeks in prečna-entropija.

Gini-jev indeks je definiran kot mera za skupno varianco po vseh K razredih [6]:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

Ni težko opaziti, da bo imel Gini-jev indeks malo vrednost, če bodo vse vrednosti $\hat{p}_{mk}$ blizu ena ali nič. Zato Gini-jevemu indesku pravimo tudi mera za čistost vozlišča – mala vrednost pomeni, da vozlišče večinoma vsebuje opazovanja iz istega razreda.

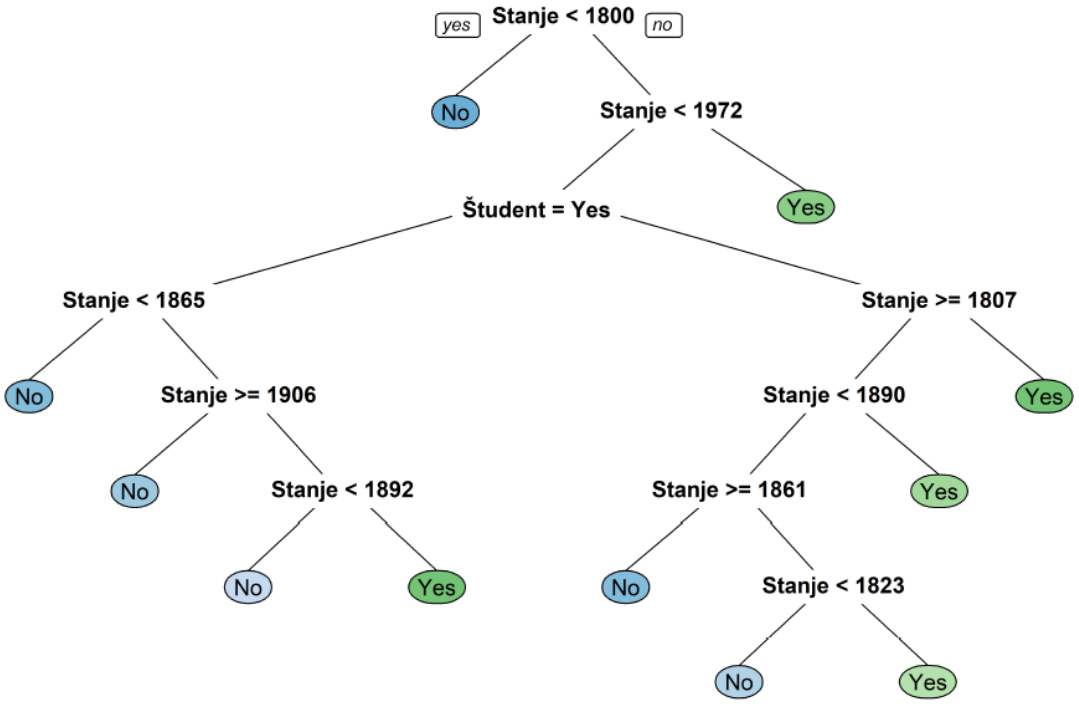
Poleg Gini-jevega indeksa se uporablja tudi prečna-entropija, ki je definirana kot [6]

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}).$$

Ker je $0 \leq \hat{p}_{mk} \leq 1$, sledi da je $-\hat{p}_{mk} \log(\hat{p}_{mk}) \geq 0$. Da se pokazati, da bo imela prečna-entropija malo vrednost, če bodo vse vrednosti $\hat{p}_{mk}$ blizu nič ali ena. Torej bo prečna-entropija majhna, če bo vozlišče čisto. Izkaže se, da sta si Gini-jev indeks in prečna-entropija numerično zelo podobna.

Na sliki 5.2 imamo primer klasifikacijskega drevesa, ki je zgrajeno na podatkih "Default" v programskem jeziku R. Želimo napovedati, če bo posameznik zmožen odplačati kredit ali ne.

Možni vrednosti sta *Yes* in *No*. Na voljo imamo vhodni spremenljivki *Stanje*, ki predstavlja stanje na računu po plačilu anuitete in *Študent*, ki pove, če je posameznik študent ali ne. Vidimo, da ni nujno, da so vse spremenljivke numerične, ampak so lahko tudi kategorične. V drevesu se pojavi tudi pogoj *Študent* = *Yes*, kar pomeni, če je posameznik študent, potem se razvije leva veja drevesa, sicer se razvije desna veja.



Slika 5.2: Klasifikacijsko drevo.

5.3 Prednosti in slabosti odločitvenih dreves

Prednosti odločitvenih dreves so:

- Odločitvena drevesa lahko enostavno razložimo.
- Nekateri ljudje trdijo, da odločitvena drevesa, v primerjavi z ostalimi pristopi, zelo odsevajo način človeškega sprejemanja odločitev.
- Odločitvena drevesa lahko predstavimo na lep grafični način.
- Odločitvena drevesa lahko brez težav obravnavajo kvalitativne spremenljivke, zato ne rabimo ustvarjati slamnatih spremenljivk.

Slabost odločitvenih dreves pa je ta, da običajno nimajo tako dobrih napovedi kot ostali pristopi za regresijo in razvrščanje. To slabost se da odpraviti, zato si bomo v nadaljevanju pogledali metodo vrečenja in naključne gozdove.

5.4 Metoda vrečenja in naključni gozdovi

Napovedne modele odločitvenih dreves lahko izboljšamo z agregacijo večjega števila dreves. Najprej si bomo podrobneje pogledali metodo vrečenja.

5.4.1 Metoda vrečenja

Metoda zankanja, ki je predstavljena v poglavju 4.4.2, je izjemno močna ideja. Pri metodi vrečenja bomo zankanje uporabili tako, da bomo z njegovo pomočjo izboljšali odločitveno drevo.

Velika pomanjkljivost odločitvenih dreves, ki so predstavljena v poglavju 5.1, je visoka varianca, kar pomeni, da če učne podatke naključno razdelimo na dva dela in zgradimo drevesi na teh podatkih, bi se lahko rezultati dobljenih dreves zelo razlikovali. Da bi zmanjšali varianco metode strojnega učenja, si pomagamo z metodo agregatnega zankanja oziroma vrečenja, ki se v praksi pogosto uporablja z odločitvenimi drevesi.

Spomnimo se, če imamo n neodvisnih spremenljivk $Z_1, \ldots, Z_n$, vsako z varianco σ^2, je varianca od povprečja $\overline{Z} = \frac{Z_1+\cdots+Z_n}{n}$ enaka σ^2/n. To pomeni, da povprečenje naključnih spremenljivk znižuje varianco. Podobno logiko lahko uporabimo na odločitvenih drevesih. Vzamemo več učnih množic in na vsaki učni množici zgradimo drevo. Za končno napoved pa vzamemo povprečno napoved zgrajenih dreves. Matematični zapis tega je [6]:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x),$$

kjer je B število učnih množic in $\hat{f}^b(x)$ napoved drevesa zgrajenega na b-ti učni množici.

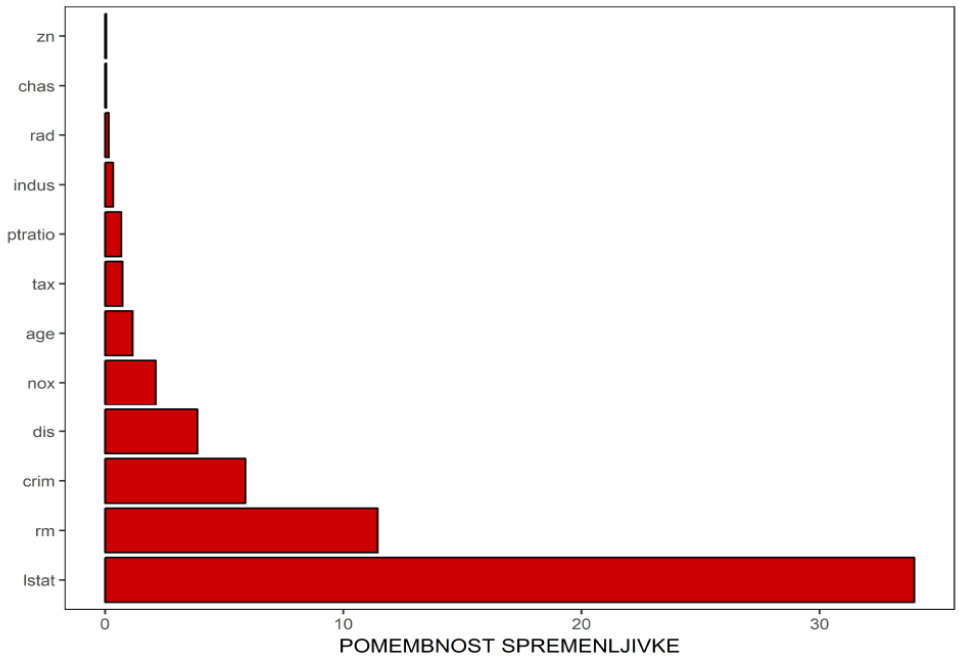
Ker v praksi običajno nikoli nimamo B različnih učnih množic, si pomagamo z zankanjem. Iz učnih podatkov z naključnim izbiranjem s ponavljanjem ustvarimo B množic, na katerih zgradimo drevesa. Tako dobimo naslednjo povprečno napoved [6]:

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

kjer je $\hat{f}^{*b}(x)$ napoved drevesa zgrajenega na b-ti množici, pridobljeni z zankanjem. To je napoved, pridobljena z metodo vrečenja, ki ima manjšo varianco ter posledično večjo natančnost.

Do zdaj smo spoznali metodo vrečenja za regresijska drevesa, zato nas zdaj zanima, kaj naredimo v primeru klasifikacijskih dreves. Obstaja veliko pristopov in eden najpreprostejših je pristop večine glasov. Tudi tukaj zgradimo drevesa na različnih množicah učnih podatkov in za končno napoved izberemo tisto, ki se največkrat pojavi.

Z uporabo metode vrečenja povečamo natančnost modela, izgubimo pa močno prednost odločitvenih dreves, tj. interpretacija rezultatov. Ker imamo več dreves, za predstavitev rezultatov ne moremo uporabiti samo enega drevesa, kot na sliki 5.1, in ni več jasno, katere spremenljivke so pri postopku odločanja najpomembnejše. Zato moramo pomembnost spremenljivk pridobiti na drug način. Spomnimo se, da drevesa gradimo s pomočjo vrednosti RSS (za regresijska drevesa) oziroma Gini-jevega indeksa (za klasifikacijska drevesa) in sicer tako, da pri vsaki delitvi pogledamo, kje se ta vrednost najbolj zmanjša. Ker imamo pri vrečenju opravka z več drevesi, si lahko pri vsakem drevesu za vsako spremenljivko shranimo spremembo vrednosti RSS. Da pri vrečenju dobimo pomembnost posamezne spremenljivke, moramo za izbrano spremenljivko skupno spremembo vrednosti RSS pri vseh drevesih deliti s številom vseh dreves. Večje kot je to povprečje, bolj je spremenljivka pomembna. Grafična predstavitev pomembnosti spremenljivk je prikazana na sliki 5.3.



Slika 5.3: Pomembnost spremenljivk za podatke *Boston*, ki so na voljo v programskem jeziku R v paketu MASS. Poskušali smo napovedati povprečno vrednost stanovanj na podeželju Bostona.

5.4.2 Naključni gozdovi

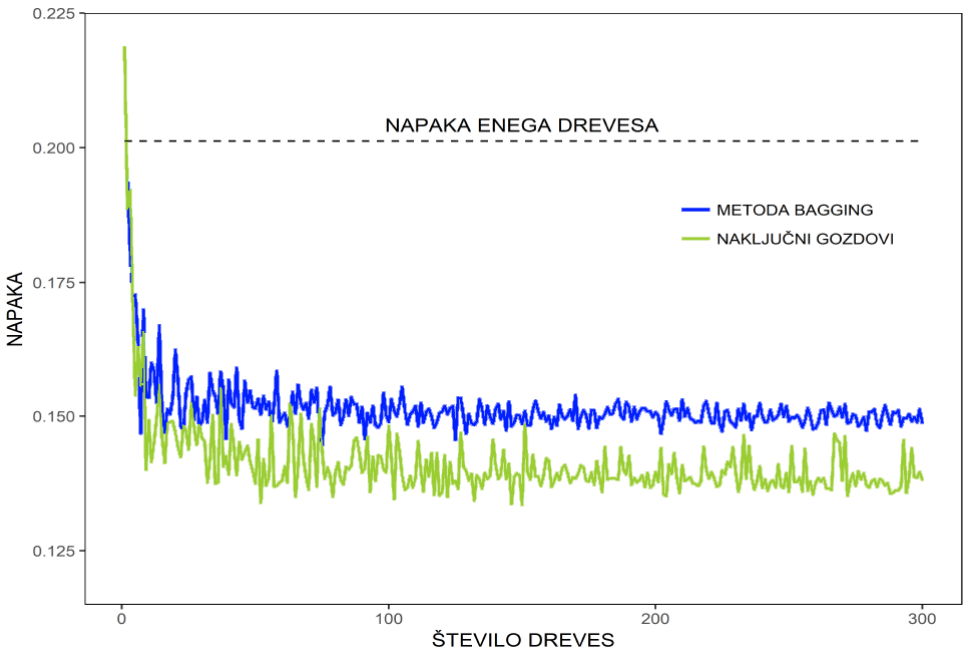
Naključni gozdovi predstavljajo izboljšavo metode vrečenja. Razlika s prejšnjo metodo je ta, da pri gradnji drevesa naredimo majhen popravek, ki dekorelira drevesa.

Še vedno s pomočjo zankanja gradimo več dreves. Razlika se pojavi pri gradnji posameznega drevesa, saj naključni gozdovi pri vsaki delitvi ne obravnavajo vseh možnih spremenljivk. Recimo, da želimo uporabiti p spremenljivk za napoved izhodne spremenljivke. Naključni gozdovi pri posamezni delitvi naredijo naključni izbor m spremenljivk ($m < p$) in ta delitev lahko upošteva samo te izbrane spremenljivke. Takšen naključni izbor se naredi za vsako delitev posebej in običajno izberemo $m \approx \sqrt{p}$.

Ta postopek se morda zdi nelogičen, vendar se izkaže, da je boljši. Recimo, da imamo spremenljivko X, ki odlično napoveduje spremenljivko Y, in nekaj spremenljivk, ki pa zmerno dobro napovedujejo Y. Z metodo vrečenja bodo vsa drevesa zgrajena podobno, saj bo prva delitev pri vseh drevesih po spremenljivki X. Zato bodo vsa drevesa izgledala zelo podobno in napovedi dreves bodo med seboj zelo korelirane. Iz statistike pa vemo, da povprečenje močno koreliranih spremenljivk ne zmanjša variance toliko kot povprečenje nekoreliranih spremenljivk. To pomeni, da v tem primeru metoda vrečenja ne bo veliko zmanjšala variance v primerjavi z enim drevesom.

Naključni gozdovi rešijo ta problem tako, da vsako delitev prisilijo, da upošteva le neko podmnožico vhodnih spremenljivk. Zato v povprečju $(p - m)/p$ delitev ne bo smelo upoštevati močne spremenljivke in tako bodo ostale imele večjo možnost, da bodo izbrane. Za prejšnji primer to pomeni, da pri nekaterih drevesih prva delitev ne bo po spremenljivki X, ampak po neki drugi.

Eno odločitveno drevo, metodo vrečenja in naključne gozdove smo uporabili na podatkih *Boston*. Natančnost posameznega modela lahko vidimo na sliki 5.4, kjer opazimo, da se je najbolje odrezal naključni gozd.



Slika 5.4: Testna napaka za eno odločitveno drevo, metodo vrečenja in naključne gozdove. Testna napaka je predstavljena kot funkcija števila dreves.

Poglavje 6

Nevronske mreže

Umetne nevronske mreže ali na kratko nevronske mreže so pomembno orodje pri strojnem učenju in temeljijo na posnemanju človeških možganov pri odločanju v nekem trenutku. Človeški možgani so sestavljeni iz približno 85 milijard nevronov, današnje umetne nevronske mreže pa le iz nekaj sto nevronov, zato je trenutno kakršnakoli primerjava med umetnimi in biološkimi nevronskimi mrežami nemogoča.

V zadnjih 50-ih letih so bile osnovne nevronske mreže uporabljene za posnemanje človeškega odločanja in reševanja problemov. To je na začetku vključevalo učenje logične IN funkcije ali učenje logične ALI funkcije. S prvimi poskusi z nevronskimi mrežami so znanstveniki želeli razumeti delovanje bioloških nevronov. Z razvojem tehnologije so napredovale tudi nevronske mreže in postale bolj kompleksne. Danes se uporabljajo za reševanje različnih problemov, od podjetniških odločitev do ekonomskih problemov in tudi za napovedovanje raznih vedenj. Nevronske mreže so pogosto orodje pri podatkovnem rudarjenju, največkrat se uporabljajo pri klasifikaciji, napovedovanju, metodi rojenja ter tudi pri [7]:

- Prepoznavanju pisave in govora.
- Avtomatiki v pametnih napravah.
- Napovedovanju vremena ter prepoznavanju podnebnih vzorcev ter veliko drugih družabnih, znanstvenih ter ekonomskih pojavov.

Nevronske mreže so najbolj uporabne na problemih, kjer so vhodni in izhodni podatki dobro definirani, povezava med njimi pa je zelo kompleksna. Zaradi odlične zmožnosti prepoznavanja vzorcev so nevronske mreže zelo priljubljene. Ostali razlogi, zakaj so nevronske mreže koristne pri podatkovnem rudarjenju, so:

1. V nasprotju z nekaterimi modeli, kot je na primer GLM, pri nevronskih mrežah nimamo predpostavk o strukturi modela in porazdelitvi podatkov. Ker je model večinoma odvisen od vzorcev oziroma karakteristik, ki se jih nevronska mreža nauči iz podatkov, so nevronske mreže zelo prilagodljive. Zato je ta način idealen za reševanje problemov iz resničnega sveta ter prepoznavanje še neodkritih vzorcev.

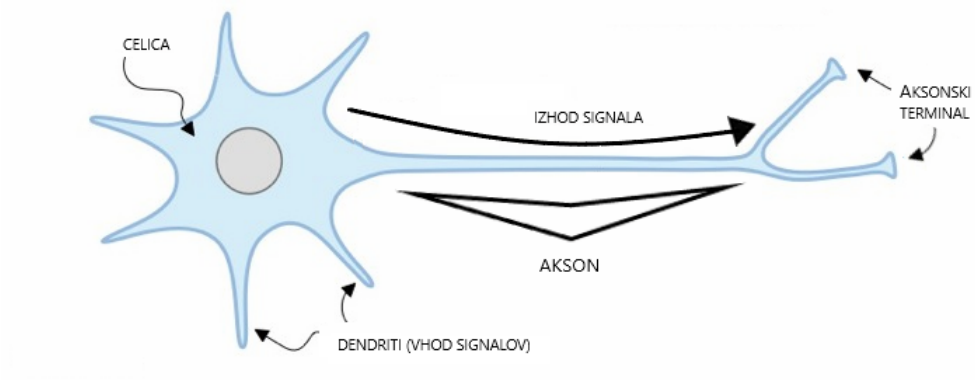
2. Matematični pristop nevronskih mrež pri predstavljanju raznih kompleksnih povezav in natančnosti aproksimacije je dobro razvit in podprt s strani teorije (Stinchcombe, Hornik in White, 1989; Cybenko, 1989; Chen in Chen, 1995). Zaradi fleksibilnosti in univerzalnosti pri modeliranju podatkov so nevronske mreže močnejše orodje kot modeli s predpisanimi pristopi. Pri problemih prepoznavanja vzorcev, napovedovanja in razvrščanja se soočamo z mapiranjem in aproksimacijo funkcije, zato je pomembno ustrezno prepoznati iskano funkcijo. Tako lahko opazimo nove povezave, ki jih s slabšimi aproksimacijami ne bi.

3. Nevronske mreže spadajo med nelinearne modele. Ker so resnični podatki mnogokrat nelinearni, tradicionalni (linearni) modeli hitro odpovedo. Nevronske mreže pa s svojo neparametričnostjo in nelinearnostjo bolje modelirajo zelo kompleksne probleme.

4. Z nevronskimi mrežami lahko rešujemo probleme, kjer imamo v podatkih veliko šuma in veliko spremenljivk.

6.1 Od bioloških do umetnih nevronskih mrež

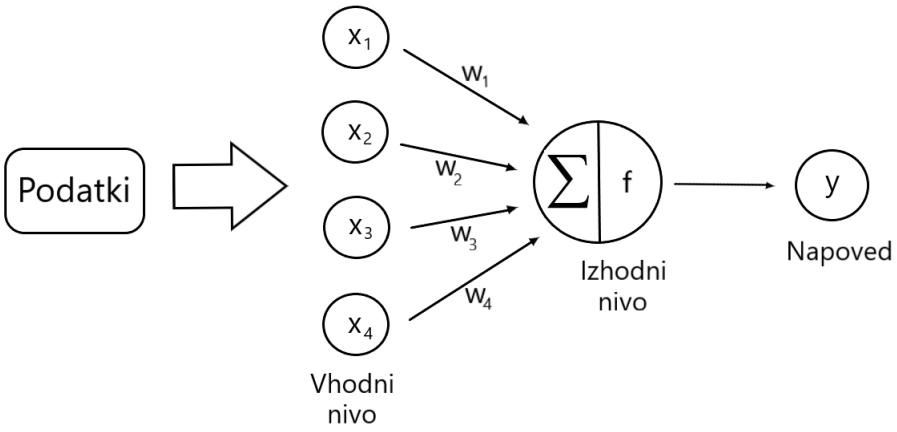
Umetne nevronske mreže temeljijo na poskusu posnemanja bioloških nevronskih mrež, zato si za boljše razumevanje poglejmo delovanje bioloških nevronov.



Slika 6.1: Biološki nevron.

Na sliki 6.1 vidimo, da celica preko dendritov sprejema vhodne signale oziroma impulze, ki jih nato uteži glede na njihovo pomembnost ali frekvenco. Celica zbira sprejete impulze in ko ti dosežejo določen prag, celica sproži izhodni signal, ki ga pošlje po aksonu do aksonskega terminala. V tem delu nevrona se električni signali pretvorijo v kemijske signale in ti so nato preko mikro-vrzeli, ki jih imenujemo sinapse, poslani do sosednjih nevronov.

Delovanje in sestava umetnega nevrona sta podobna biološkemu. Kot vidimo na sliki 6.2, usmerjeni nevronski diagram definira povezavo med vhodnimi signali (spremenljivke x_i), ki so sprejeti preko dendritov, in izhodnim signalom (spremenljivka y). Podobno kot pri biološkem nevronu tudi tukaj vse vhodne signale utežimo (vrednosti w_i) glede na njihovo pomembnost. Uteženi vhodni signali so nato sešteti v telesu celice in izhodni signal je poslan naprej glede na aktivacijsko funkcijo (funkcija f).



Slika 6.2: Umetni nevron.

Običajno umetni nevron, ki ima n vhodnih podatkov, matematično predstavimo s formulo [7]

$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right).$$

Utež w_i omogoča povečanje ali zmanjšanje vpliva signala x_i na skupno vsoto vhodnih signalov. To vsoto nato uporabi aktivacijska funkcija f, ki nam določi izhodni signal $y(x)$.

Nevronske mreže uporabljajo te umetne nevrone za gradnjo kompleksnega modela. Za definicijo umetne nevronske mreže so potrebne tri stvari [7]:

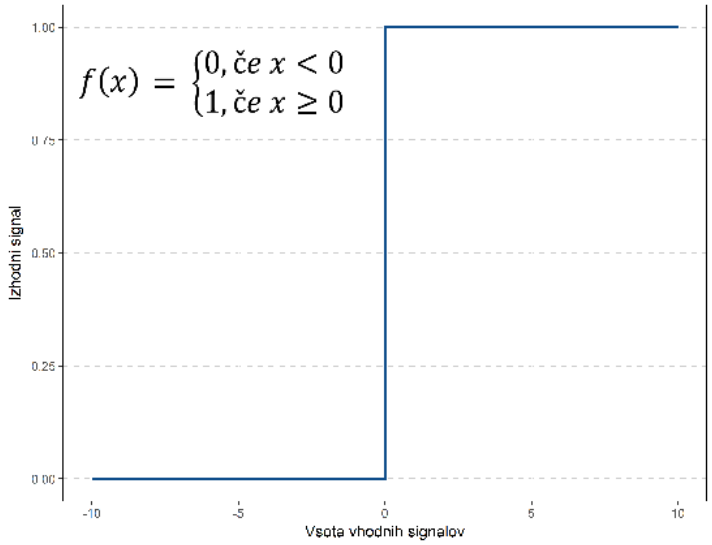
- Aktivacijska funkcija, ki transformira uteženo vsoto vhodnih signalov v izhodni signal, katerega lahko pošljemo naprej v mrežo.

- Mrežna arhitektura, ki nam poleg števila nevronov v mreži, pove tudi število nivojev in kako so ti med seboj povezani.
- Algoritem učenja, ki določa kako izračunamo uteži na povezavah med nevroni.

Poglejmo si posamezne dele bolj podrobno.

6.2 Aktivacijska funkcija

Aktivacijska funkcija je mehanizem, s katerim umetni nevron procesira vhodne podatke in jih prenese naprej po mreži. Pri bioloških nevronih je aktivacijska funkcija lahko zamišljena kot proces, ki sešteva vhodne signale in če je vsota uteženih signalov večja od določenega praga, nevron signal pošlje naprej po mreži, sicer ne naredi ničesar. V terminologiji nevronskih mrež je to znano kot aktivacijska funkcija s pragom, saj izhodni signal nastane le v primeru, ko je dosežen določen prag. Slika 6.3 prikazuje primer tipične funkcije s pragom, ki ji pravimo stopničasta aktivacijska funkcija.



Slika 6.3: Stopničasta aktivacijska funkcija.

Čeprav je aktivacijska funkcija s pragom zanimiva zaradi paralel z biologijo, je redko uporabljena v umetnih nevronskih mrežah. Aktivacijsko funkcijo izberemo na podlagi njene zmožnosti demonstriranja želenih matematičnih karakteristik in natančnega modeliranja povezav med podatki.

Obstaja veliko možnih izbir za aktivacijsko funkcijo, a največkrat se uporabljajo naslednje:

- Logistična sigmoidna funkcija:

$$f(x) = \frac{1}{1+e^{-x}}.$$

- Linearna funkcija:

$$f(x) = x.$$

- Popravljena linearna funkcija:

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}.$$

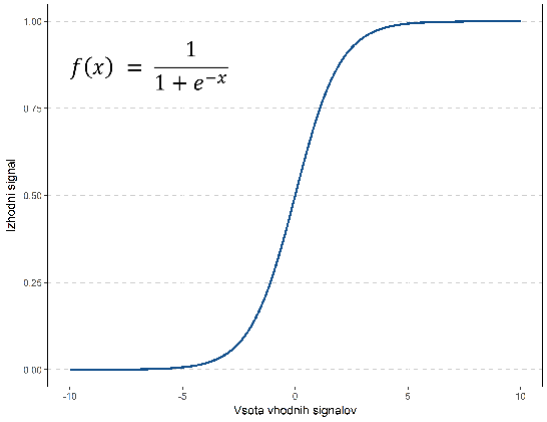
- Hiperbolični tangens:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

- Gaussova funkcija:

$$f(x) = e^{-x^2}.$$

V praksi se največkrat uporablja logistična sigmoidna aktivacijska funkcija. Iz slike 6.4 vidimo, da ima podobno S obliko kot stopničasta aktivacijska funkcija, vendar izhodni signal ni binaren in lahko zavzame poljubno vrednost med 0 in 1. Logistična sigmoidna aktivacijska funkcija je tudi odvedljiva, zato lahko izračunamo odvod funkcije na celotnem obsegu vhodnih podatkov. Kot bomo videli, je ta lastnost pomembna pri algoritmu učenja.



Slika 6.4: Logistična sigmoidna aktivacijska funkcija.

Primarna razlika med zgoraj naštetimi aktivacijskimi funkcijami je razpon izhodnega signala. Običajno je eden izmed teh: $(0, 1)$, $(-1, 1)$ ali $(-\infty, \infty)$. S pravilno izbiro aktivacijske funkcije lahko dobimo nevronsko mrežo, ki se bolje prilega določenim podatkom. Na

primer, če izberemo linearno aktivacijsko funkcijo, lahko z nevronsko mrežo dobimo zelo podobne rezultate kot z linearnim modelom.

Opazimo lahko, da je pri večini aktivacijskih funkcij obseg izhodnih signalov, na katerega imajo vpliv vhodni podatki, relativno majhen. Pri logistični sigmoidni funkciji je izhodni signal zelo blizu 0 ali 1 za vhodni signal, ki je manjši od -5 ali večji od 5. Za zelo dinamične vhodne podatke lahko takšno stiskanje signala privede do nasičenega signala pri visoki ali nizki vrednosti izhodnega signala. Ta problem lahko odpravimo s transformacijo vhodnih podatkov, tako da njihove vrednosti padejo v majhen interval okoli 0. To običajno dosežemo z normalizacijo ali standardizacijo vhodnih podatkov. Tako bo aktivacijska funkcija imela akcijo čez celoten razpon in spremenljivke z velikimi vrednostmi, kot sta na primer masa vozila in moč vozila, ne bodo prevladovale spremenljivk z majhnimi vrednostmi, kot sta na primer starost zavarovanca in starost vozila. Stranska korist tega je lahko hitrejše učenje modela, saj lahko algoritem hitreje iterira čez akcijski razpon vhodnih podatkov.

6.3 Mrežna arhitektura

Zmožnost učenja nevronske mreže je odvisna od njene arhitekture ali od vzorcev in strukture povezanih nevronov. Čeprav obstaja ogromno mrežnih arhitektur, jih lahko razlikujemo po treh glavnih lastnostih:

- število nivojev,
- ali lahko informacije v mreži potujejo nazaj,
- število vozlišč znotraj posameznega nivoja.

Z mrežno arhitekturo določimo kompleksnost problemov, ki jih lahko rešimo z nevronsko mrežo. Običajno lahko večje in kompleksnejše nevronske mreže bolje prepoznajo skrite vzorce, vendar je njihovo učenje in delovanje počasnejše.

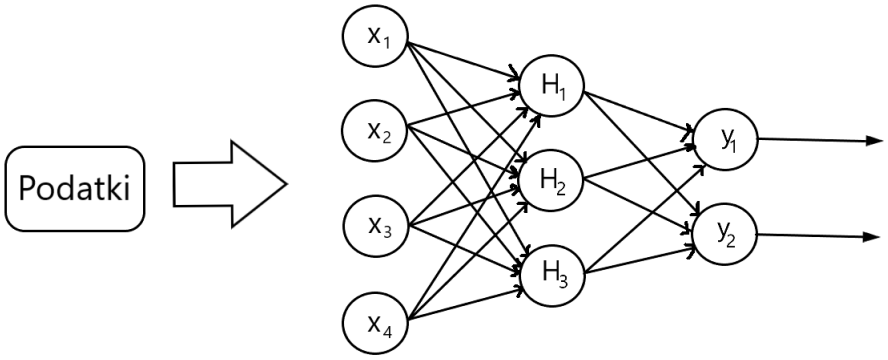
6.3.1 Število nivojev

Za definicijo mrežne arhitekture potrebujemo terminologijo, ki razlikuje umetne nevrone glede na njihov položaj v nevronski mreži. Na sliki 6.2 je primer arhitekture eno-nivojske nevronske mreže. Množica nevronov, imenovanih vhodna vozlišča, prejme neprocesirane signale neposredno iz vhodnih podatkov. Vsako vhodno vozlišče je zadolženo za procesiranje

ene spremenljivke med podatki. Vrednost spremenljivke se transformira glede na aktivacijsko funkcijo v pripadajočem vozlišču. Nato signale, poslane iz vhodnih vozlišč, prejme izhodno vozlišče, ki uporabi svojo aktivacijsko funkcijo za generiranje končne napovedi.

Vhodna in izhodna vozlišča so razdeljena v skupine, ki jih imenujemo nivoji. Ker vhodna vozlišča procesirajo vhodne podatke natanko take, kot so v originalu, ima mreža na sliki 6.2 samo eno množico uteži, označenih z w_1, w_2, w_3 in w_4. Zato jo imenujemo eno-nivojska nevronska mreža in uporabimo jo lahko za klasifikacijo enostavnih vzorcev, še posebej za vzorce, ki so linearno ločljivi. Vendar so za večino problemov potrebne kompleksnejše nevronske mreže.

Očitno je, da lahko zgradimo kompleksnejše nevronske mreže z dodajanjem dodatnih nivojev. Takšne nevronske mreže imenujemo večnivojske nevronske mreže, dodatnim nivojem pa pravimo skriti nivoji. Skriti nivoji obdelajo signale iz vhodnih vozlišč, preden se ti pošljejo do izhodnega vozlišča. Ni nujno, da imamo le eno izhodno vozlišče, lahko jih imamo tudi več. Večina večnivojskih nevronskih mrež je popolno povezanih, kar pomeni, da je vsako vozlišče v enem nivoju povezano z vsakim vozliščem v naslednjem nivoju, vendar to ni obvezno. Na sliki 6.5 imamo primer večnivojske nevronske mreže.



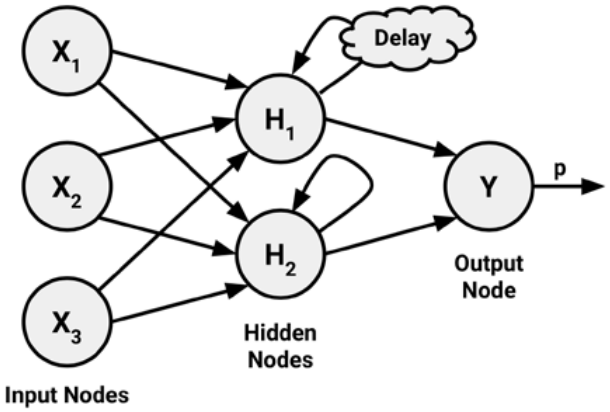
Slika 6.5: Večnivojska nevronska mreža z enim skritim nivojem in dvema izhodnima vozliščema.

6.3.2 Smer potovanja informacij

Na slikah 6.2 in 6.5 smo s puščicami nakazali, da signali potujejo samo v eno smer. Nevronske mreže, v katerih vhodni signal potuje samo v eno smer, dokler ne doseže izhodnega nivoja, imenujemo tudi feedforward mreže.

Čeprav je pretok informacij v takšnih mrežah omejen, so feedforward mreže presenetljivo fleksibilne. Na primer, spreminjamo lahko število nivojev in število vozlišč v posameznem nivoju, hkrati pa lahko modeliramo več izhodnih spremenljivk. Nevronska mreža z več skritimi nivoji se imenuje globoka nevronska mreža in učenju takšne mreže pravimo globoko učenje.

V nasprotju s feedforward mrežo pa feedbackward mreža omogoča potovanje signalov v obe smeri s pomočjo zank. Ta lastnost, ki bolje posnema delovanje biološkega nevrona, omogoča učenje zelo kompleksnih vzorcev. Dodatek kratkoročnega spomina oziroma zamika zelo poveča moč feedbackward mreže. Predvsem to vključuje zmožnost razumevanja zaporedja dogodkov čez neko časovno obdobje. To bi lahko na primer uporabili za napovedovanje na borzi in za napovedovanje vremena. Na sliki 6.6 je enostavni primer feedbackward mreže.



Slika 6.6: Feedbackward nevronska mreža.

Čeprav imajo feedbackward mreže ogromno potenciala, so še zelo teoretične in redko uporabljene v praksi. Medtem pa se feedforward mreže precej uporabljajo na problemih resničnega sveta. Dejansko so večnivojske feedforward mreže, ki jim včasih pravimo tudi Multilayer Perceptron (MLP), standardna mrežna arhitektura.

6.3.3 Število nevronov v posameznem nivoju

Zraven števila nivojev in smeri potovanja informacij se nevronske mreže razlikujejo tudi po številu vozlišč v posameznem nivoju. Število vhodnih vozlišč je določeno s številom vhodnih spremenljivk. Podobno je število izhodnih vozlišč določeno s številom izhodnih spremenljivk, ki jih želimo modelirati. Število skritih vozlišč pa lahko poljubno določimo pred učenjem modela.

Žal ne obstaja zanesljivo pravilo za določitev števila nevronov v skritih nivojih. Ustrezna izbira je odvisna od števila vhodnih spremenljivk, števila učnih podatkov, šuma v podatkih, želene kompleksnosti učenja in drugih faktorjev.

Na splošno velja, da kompleksnejše mrežne arhitekture z velikim številom mrežnih povezav omogočajo učenje kompleksnejših problemov. Veliko število nevronov lahko privede do modela, ki se zelo dobro prilagaja učnim podatkom, ne pa nujno tudi testnim podatkom. Velike nevronske mreže so lahko tudi računsko precej zahtevne in počasne za učenje. Najboljše vodilo je uporaba najmanjšega števila vozlišč, ki dajo zadovoljive rezultate na testnih podatkih.

6.4 Učenje nevronske mreže z vzvratnim razširjanjem

Mrežna arhitektura deluje po principu učenja z izkušnjami. Ko nevronska mreža procesira vhodne podatke, se povezave med nevroni krepijo ali slabijo. Uteži v nevronskih mrežah se prilagajajo, da odražajo vzorce, opazovane skozi čas.

Učenje nevronske mreže s prilagajanjem uteži je računsko zelo zahtevno. Čeprav so umetne nevronske mreže proučevali že desetletja prej, so jih za reševanje problemov resničnega sveta začeli uporabljati šele v sredini 80-ih let, ko so odkrili učinkovito metodo učenja. Algoritem je uporabljal strategijo vzvratnega razširjenja napak.

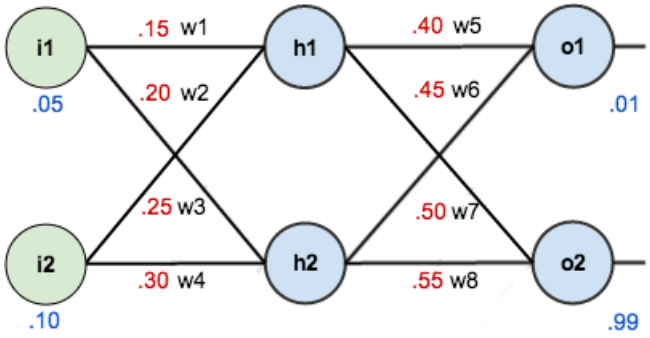
V najbolj splošni obliki algoritem z vzvratnim razširjanjem popravlja uteži tako, da ponavlja naslednja dva koraka:

- Korak naprej.
  V tem koraku so nevroni aktivirani zaporedoma od vhodnega nivoja do izhodnega nivoja. Spotoma so uporabljene posamezne nevronske uteži in aktivacijske funkcije. Ko je dosežen zadnji nivo, se proizvede izhodni signal.

- Korak nazaj.
  V tem koraku se izhodni signal, izračunan v koraku naprej, primerja s pravo ciljno vrednostjo v učnih podatkih. Razlika med pravo in napovedano vrednostjo privede do napake, ki se pošlje nazaj po mreži, da se spremenijo uteži med nevroni in zmanjšajo prihodnje napake.

Ker mreža na začetku nima predhodnega znanja, se začetne uteži običajno določijo naključno. Nato algoritem ponavlja zgornja koraka, dokler ne doseže zaustavitvenega pogoja.

Čez čas mreža uporabi informacije, ki smo jih poslali nazaj, da zmanjša skupno napako nevronske mreže. Ker je povezava med vsakim nevronskim vhodom in izhodom kompleksna, se pojavi vprašanje, kako algoritem določi, za koliko se naj utež spremeni. Odgovor na to vprašanje vključuje tehniko, imenovano gradientni spust. Algoritem vzratnega razširjanja uporabi odvod vsake nevronske aktivacijske funkcije, da prepozna naklon v smeri vsake prihajajoče uteži, zato je pomembno, da je aktivacijska funkcija odvedljiva. Naklon nakazuje, kako močno se bo napaka povečala ali zmanjšala pri spremembi uteži. Algoritem bo poskušal spremeniti uteži tako, da bo prišlo do največjega zmanjšanja napake glede na količino, znano kot stopnja učenja. Večja kot je stopnja učenja, hitreje se bo algoritem poskušal spustiti po naklonih navzdol, kar lahko zmanjša čas učenja, vendar se lahko zgodi, da najmanjšo napako preskočimo. Opisan postopek je v praksi enostavno izvedljiv. Za boljše razumevanje si poglejmo naslednji primer:



Slika 6.7: Nevronska mreža s podatki.

Na sliki 6.7 imamo nevronsko mrežo, ki je sestavljena iz dveh vhodnih nevronov, enega skritega nivoja z dvema nevronoma in dveh izhodnih nevronov. Izračunati želimo nove uteži, ob predpostavki, da sta vhodna podatka $i_1 = 0.05$ in $i_2 = 0.10$ ter izhodna podatka$o_1 = 0.01$ in $o_2 = 0.99$.

Korak naprej: V tem koraku izračunamo izhodni signal za stare uteži, ki jih razberemo iz slike 6.7. Dobimo naslednje rezultate:

$$in_{h_1} = w_1 \cdot i_1 + w_2 \cdot i_2 = 0.15 \cdot 0.05 + 0.2 \cdot 0.1 = 0.0275,$$

$$in_{h_2} = w_3 \cdot i_1 + w_4 \cdot i_2 = 0.25 \cdot 0.05 + 0.3 \cdot 0.1 = 0.0425.$$

Za nevronsko aktivacijsko funkcijo f izberemo logistično sigmoidno funkcijo $f(x) = \frac{1}{1+e^{-x}}$:

$$out_{h_1} = f(in_{h_1}) = f(0.0275) = 0.5069,$$

$$out_{h_2} = f(in_{h_2}) = f(0.0425) = 0.5106.$$

Postopek ponovimo za o_1 in o_2:

$$in_{o_1} = w_5 \cdot out_{h_1} + w_6 \cdot out_{h_2} = 0.4 \cdot 0.5069 + 0.45 \cdot 0.5106 = 0.4325,$$

$$in_{o_2} = w_7 \cdot out_{h_1} + w_8 \cdot out_{h_2} = 0.5 \cdot 0.5069 + 0.55 \cdot 0.5106 = 0.5343,$$

$$out_{o_1} = f(0.4325) = 0.6065,$$

$$out_{o_2} = f(0.5343) = 0.6305.$$

Zdaj izračunamo še skupno napako modela E_s, ki jo izračunamo kot $E_s = \frac{1}{2}\sum_{i=1}^{2}(o_i - out_{o_i})^2$. V našem primeru dobimo naslednji rezultat:

$$\begin{aligned} E_{o_1} &= \frac{1}{2}(o_1 - out_{o_1})^2 = \frac{1}{2}(0.01 - 0.6065)^2 = 0.1779, \\ E_{o_2} &= \frac{1}{2}(o_2 - out_{o_2})^2 = \frac{1}{2}(0.99 - 0.6305)^2 = 0.0646, \\ E_s &= E_{o_1} + E_{o_2} = 0.1779 + 0.0646 = 0.2425. \end{aligned}$$

Korak nazaj: V tem koraku s pomočjo gradienta spremenimo vse uteži, začnemo z utežmi w_5, w_6, w_7 in w_8. Najprej nas zanima, koliko vpliva sprememba uteži w_5 na skupno napako E_s oziroma $\frac{\partial E_s}{\partial w_5}$. Če upoštevamo verižno pravilo, dobimo naslednje:

$$\frac{\partial E_s}{\partial w_5} = \frac{\partial E_s}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial in_{o_1}} \cdot \frac{\partial in_{o_1}}{\partial w_5}.$$

Poračunamo posamezne dele:

- $\frac{\partial E_s}{\partial out_{o_1}}$:

$$\begin{aligned} E_s &= \frac{1}{2}(o_1 - out_{o_1})^2 + \frac{1}{2}(o_2 - out_{o_2})^2, \\ \frac{\partial E_s}{\partial out_{o_1}} &= (o_1 - out_{o_1}) \cdot (-1) + 0 = -(0.01 - 0.6065) = 0.5965. \end{aligned}$$

- $\frac{\partial out_{o_1}}{\partial in_{o_1}}$:

$$\begin{aligned} out_{o_1} &= f(in_{o_1}) = \frac{1}{1 + e^{-in_{o_1}}}, \\ \frac{\partial out_{o_1}}{\partial in_{o_1}} &= -(1 + e^{-in_{o_1}})^{-2} \cdot e^{-in_{o_1}} \cdot (-1) = (1 + e^{-0.4325})^{-2} \cdot e^{-0.4325} = 0.5668. \end{aligned}$$

- $\frac{\partial in_{o_1}}{\partial w_5}$:

$$in_{o_1} = w_5 \cdot out_{h_1} + w_6 \cdot out_{h_2},$$
$$\frac{\partial in_{o_1}}{\partial w_5} = out_{h_1} + 0 = 0.5069.$$

Če posamezne dele zdaj združimo, dobimo

$$\frac{\partial E_s}{\partial w_5} = \frac{\partial E_s}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial in_{o_1}} \cdot \frac{\partial in_{o_1}}{\partial w_5} = 0.5965 \cdot 0.5668 \cdot 0.5069 = 0.1714.$$

Da zmanjšamo napako, moramo od trenutne uteži w_5 odšteti zgoraj izračunano vrednost, pomnoženo s stopnjo učenja η, ki bo v našem primeru $\eta = 0.5$. Tako dobimo novo utež w_5^+, ki je

$$w_5^+ = w_5 - \eta \cdot \frac{\partial E_s}{\partial w_5} = 0.4 - 0.5 \cdot 0.1714 = 0.3143.$$

Po analognem postopku dobimo nove uteži w_6^+, w_7^+ in w_8^+:

$$w_6^+ = 0.45 - 0.5 \cdot 0.1726 = 0.3637,$$
$$w_7^+ = 0.5 - 0.5 \cdot (-0.0425) = 0.5213,$$
$$w_8^+ = 0.55 - 0.5 \cdot (-0.0428) = 0.5714.$$

Posodobitev uteži v nevronski mreži izvedemo šele, ko imamo izračunane vse nove uteži, torej tudi uteži w_1^+, w_2^+, w_3^+ in w_4^+. Zato moramo izračunati še nove uteži za prvi del nevronske mreže, pri tem pa uporabimo stare uteži w_5, w_6, w_7, w_8, in ne novih.

Spet upoštevamo verižno pravilo:

$$\frac{\partial E_s}{\partial w_1} = \frac{\partial E_s}{\partial out_{h_1}} \cdot \frac{\partial out_{h_1}}{\partial in_{h_1}} \cdot \frac{\partial in_{h_1}}{\partial w_1}.$$

Poračunamo posamezne dele:

- $\frac{\partial E_s}{\partial out_{h_1}}$:

  Ker out_{h_1} vpliva na out_{o_1} in out_{o_2}, dobimo

$$E_s = E_{o_1} + E_{o_2},$$
$$\frac{\partial E_s}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}}.$$

Najprej izračunajmo $\frac{\partial E_{o_1}}{\partial out_{h_1}}$:

$$\begin{aligned}\frac{\partial E_{o_1}}{\partial out_{h_1}} &= \frac{\partial E_{o_1}}{\partial in_{o_1}} \cdot \frac{\partial in_{o_1}}{\partial out_{h_1}},\\ in_{o_1} &= w_5 \cdot out_{h_1} + w_6 \cdot out_{h_2},\\ \frac{\partial in_{o_1}}{\partial out_{h_1}} &= w_5 = 0.40.\end{aligned}$$

Za izračun $\frac{\partial E_{o_1}}{\partial in_{o_1}}$ lahko uporabimo že prej izračunane vrednosti:

$$\begin{aligned}E_{o_1} &= \frac{1}{2}(o_1 - out_{o_1})^2,\\ \frac{\partial E_{o_1}}{\partial in_{o_1}} &= \frac{\partial E_{o_1}}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial in_{o_1}} = -(o_1 - out_{o_1}) \cdot \frac{\partial out_{o_1}}{\partial in_{o_1}} = 0.5965 \cdot 0.5668 = 0.3381.\end{aligned}$$

Vstavimo te vrednosti in dobimo

$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = 0.3381 \cdot 0.4 = 0.1352.$$

Postopek ponovimo za $\frac{\partial E_{o_2}}{\partial out_{h_1}}$ in dobimo

$$\frac{\partial E_{o_2}}{\partial out_{h_1}} = -0.0419.$$

Skupaj dobimo

$$\frac{\partial E_s}{\partial out_{h_1}} = 0.1352 + (-0.0419) = 0.0933.$$

- $\frac{\partial out_{h_1}}{\partial in_{h_1}}$:

$$\begin{aligned}out_{h_1} &= f(in_{h_1}) = \frac{1}{1 + e^{-in_{h_1}}},\\ \frac{\partial out_{h_1}}{\partial in_{h_1}} &= -(1 + e^{-in_{h_1}})^{-2} \cdot e^{-in_{h_1}} \cdot (-1) = (1 + e^{-0.0275})^{-2} \cdot e^{-0.0275} = 0.2500.\end{aligned}$$

- $\frac{\partial in_{h_1}}{\partial w_1}$:

$$\begin{aligned}in_{h_1} &= w_1 \cdot i_1 + w_2 \cdot i_2,\\ \frac{\partial in_{h_1}}{\partial w_1} &= i_1 = 0.05.\end{aligned}$$

Vse skupaj združimo:

$$\frac{\partial E_s}{\partial w_1} = 0.0933 \cdot 0.25 \cdot 0.05 = 0.0012.$$

Posodobimo utež w_1:

$$w_1^+ = w_1 - \eta \cdot \frac{\partial E_s}{\partial w_1} = 0.15 - 0.5 \cdot 0.0012 = 0.1494.$$

Postopek ponovimo tudi za uteži w_2, w_3 in w_4:

$$w_2^+ = 0.2 - 0.5 \cdot 0.0023 = 0.1989,$$
$$w_3^+ = 0.25 - 0.5 \cdot 0.0022 = 0.2489,$$
$$w_4^+ = 0.3 - 0.5 \cdot 0.0004 = 0.2998.$$

Ko posodobimo vse uteži, lahko izračunamo napoved in napako z novimi utežmi. Novi napovedi sta:

$$out_{o_1}^+ = 0.585406169,$$
$$out_{o_2}^+ = 0.635520715.$$

Prej je bila napaka enaka 0.2425, sedaj pa je enaka 0.2284. Vidimo, da se napaka sicer ni zmanjšala za veliko, vendar moramo upoštevati, da smo naredili le eno iteracijo. Če to ponovimo večkrat, na primer 10000-krat, se bo napaka zmanjšala in imeli bomo boljše napovedi za izhodne spremenljivke.

Poglavje 7

Predstavitev podatkov

V tem poglavju bomo predstavili praktični del magistrskega dela. Posplošeni linearni model, naključni gozd in nevronsko mrežo bomo uporabili za napovedovanje višine škode pri avtomobilskem zavarovanju. Podatki, ki smo jih uporabili, so dostopni preko paketa CASdatasets2016 za programski jezik R pod imenom freMTPL.

7.1 Spremenljivke

Prvotno so podatki sestavljeni iz 11-ih spremenljivk, ki jih lahko vidimo v tabeli 7.1. Vsaka vrstica v tabeli podatkov predstavlja natanko eno polico.

Spremenljivka	Tip	Opis
Pogodba	Numerična	Številka police.
Stevilo_skod	Numerična	Število škod v času izpostavljenosti.
Visina_skode	Numerična	Skupna višina škod v času izpostavljenosti.
Izpostavljenost	Numerična	Izpostavljenost police v letih.
Moc_motorja	Kategorična	Moč motorja vozila.
Starost_voznika	Numerična	Starost voznika v letih.
Starost_vozila	Numerična	Starost vozila v letih.
Znamka_vozila	Kategorična	Znamka vozila.
Gorivo	Kategorična	Gorivo vozila.
Regija	Kategorična	Regija bivanja voznika.
Gostota_prebivalstva	Numerična	Gostota prebivalstva na km^2 v kraju bivanja voznika.

Tabela 7.1: Spremenljivke.

7.2 Obdelava podatkov

Pri obdelavi podatkov smo imeli tri glavne cilje:

1. Začetna obdelava.
   Razumeti in odpraviti potencialne napake v podatkih, ki bi lahko ovirale izvedbo strojnega učenja.

2. Množice podatkov.
   Pri metodah strojnega učenja moramo določiti izhodno spremenljivko, ki jo želimo napovedati in vhodne spremenljivke, ki jih bomo obravnavali. Prav tako potrebujemo delitev na učne in testne podatke.

3. Raziskovalna analiza podatkov.
   Pridobiti začetno razumevanje o pomembnih trendih in vzorcih, ki lahko zahtevajo dodatno analizo.

Ti koraki so podrobno opisani v nadaljevanju.

7.2.1 Začetna obdelava

Po začetnem pregledu podatkov smo ugotovili, da so le-ti konsistentni in ne vsebujejo napačnih vnosov, kot bi lahko bila na primer negativna višina škode.

Za uspešno izvedbo strojnega učenja smo kategorične spremenljivke pretvorili v numerične. Moč motorja smo iz črk pretvorili v števila od 1 do 12, kjer večje število pomeni močnejši motor, da je računalnik lahko bolje interpretiral vpliv moči motorja. Ker preostalih kategoričnih spremenljivk nismo mogli smiselno pretvoriti v številke, smo jih nadomestili z binarnimi spremenljivkami. Tako smo spremenljivko *Gorivo*, ki zavzema vrednosti *dizel* in *bencin*, nadomestili s spremenljivkama *Gorivo_dizel*, ki ima vrednost 1 pri vozilih na dizel ter 0 pri vozilih na bencin, in *Gorivo_bencin*, ki ima vrednost 1 pri vozilih na bencin ter 0 pri vozilih na dizel. Podobno smo naredili za preostale kategorične spremenljivke.

7.2.2 Množice podatkov

Po obliki so vhodni podatki že primerni za delitev na učne in testne podatke, ki jih lahko uporabimo za strojno učenje. Pri tem je naša ciljna spremenljivka, ki jo želimo napovedati, *Visina_skode*, vhodne spremenljivke pa so *Moc_motorja*, *Starost_voznika*, *Starost_vozila*,

Znamka_vozila, *Gorivo*, *Regija* in *Gostota_prebivalstva*. Preostale spremenljivke za napovedovanje višine škode niso primerne.

Na tem koraku je pomembno, kako razdelimo podatke na učne in testne podatke. Obstaja več možnosti, kako lahko to naredimo. Odločili smo se za naključno razdelitev podatkov, kar pomeni, da je bila posamezna polica naključno dodeljena učni ali testni množici, pri tem je bila verjetnost dodelitve v učno množico enaka 0.80.

7.2.3 Raziskovalna analiza podatkov

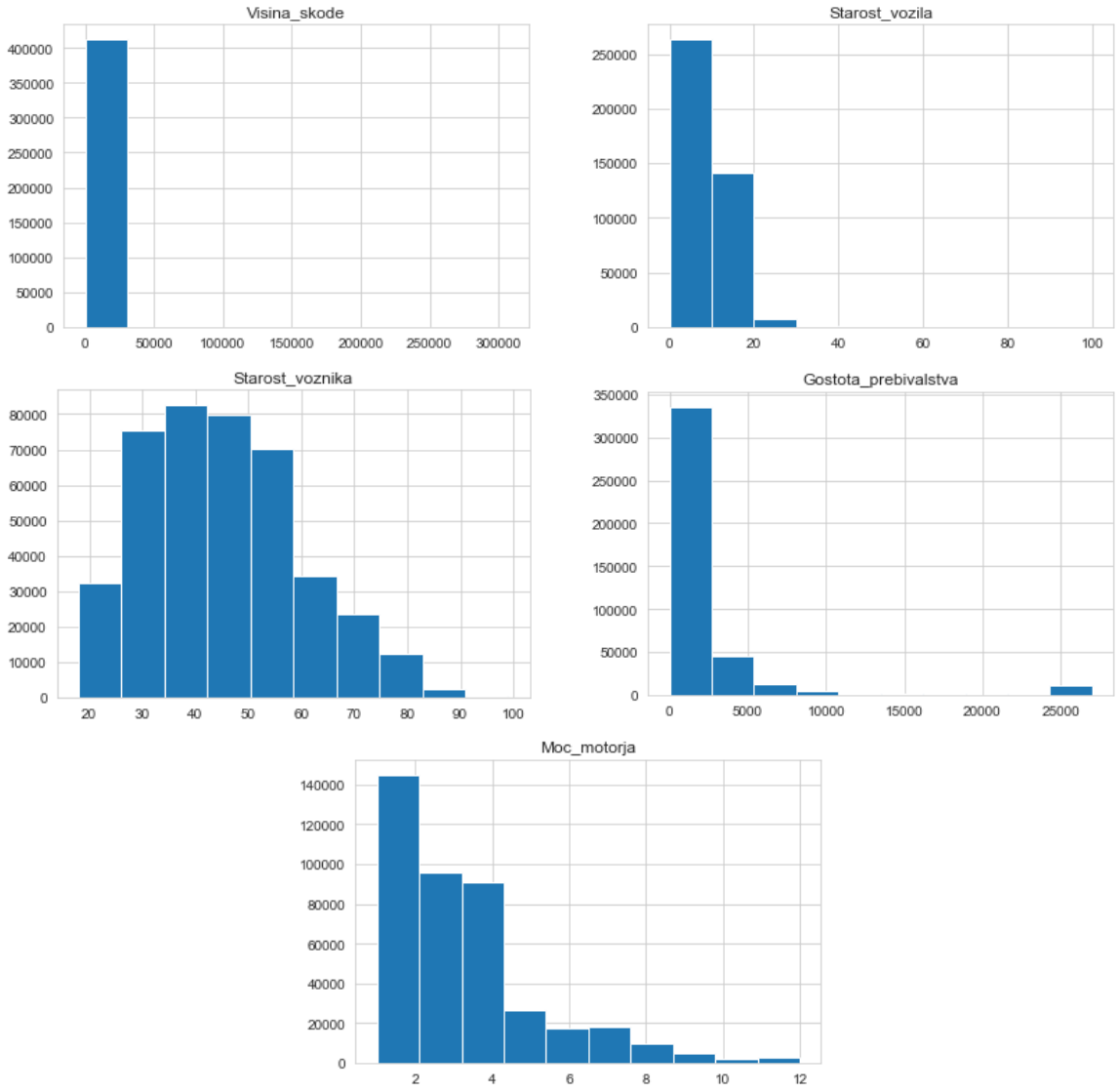
Raziskovalna analiza je vsebovala analizo porazdelitve posamezne spremenljivke, analizo osamelcev in analizo korelacije med spremenljivkami.

Za vsako obravnavano numerično spremenljivko smo porazdelitev analizirali numerično, tako da smo pogledali razpon, in grafično, s pomočjo histograma. Kategorične spremenljivke smo analizirali samo grafično, s stolpičnim grafom.

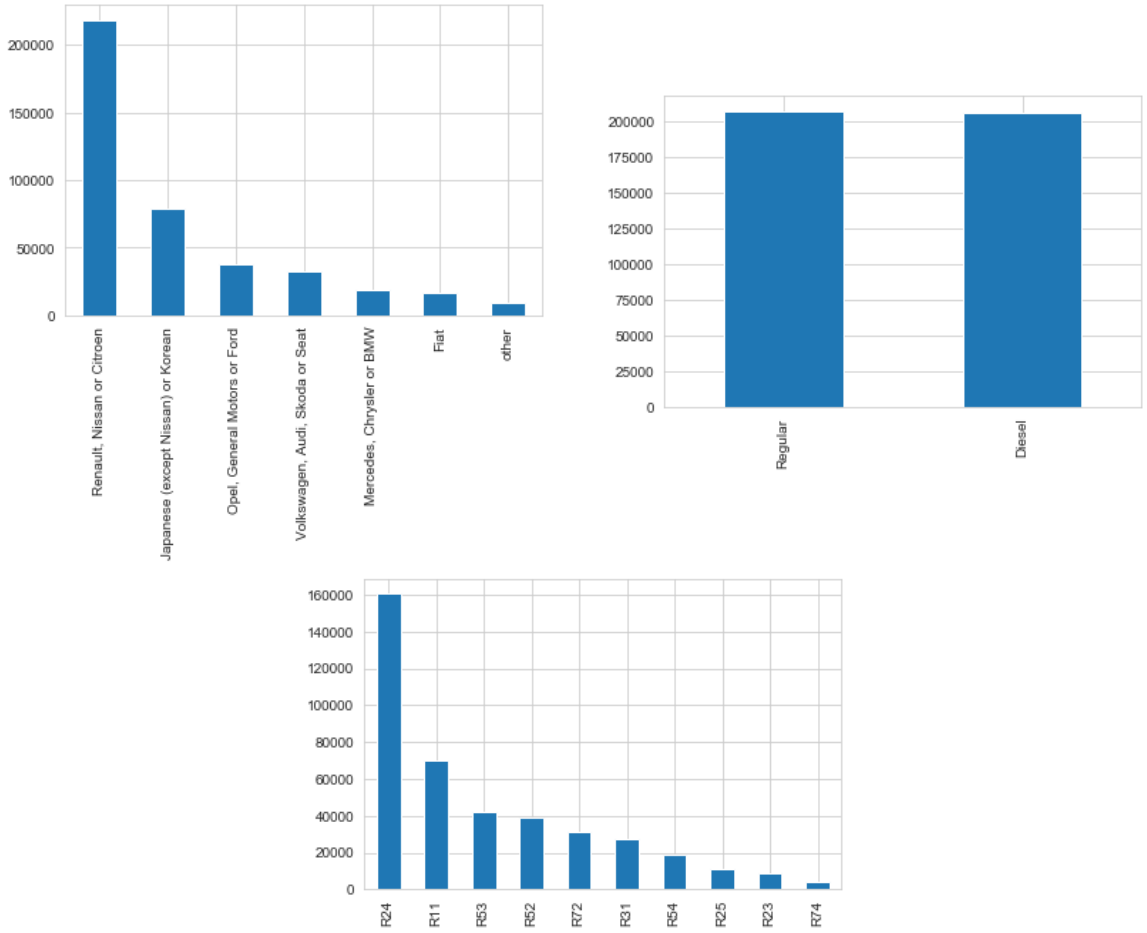
Na podlagi rezultatov, ki so na slikah 7.1 in 7.2, lahko sklepamo, da so porazdelitve numeričnih spremenljivk asimetrične v desno, kar pomeni, da imamo opravka z večjim številom manjših škod, novejših avtomobilov, mlajših voznikov, manj poseljenih krajev in šibkejših motorjev. Na sliki 7.3 pa lahko vidimo, da so najbolj zastopana vozila znamk Renault, Nissan in Citroen. Vozil na bencin je približno toliko kot vozil na dizel. Največ voznikov pa prihaja iz regij R24 in R11.

	Visina_skode	Moc_motorja	Starost_vozila	Starost_voznika	Gostota_prebivalstva
count	413167.0	413167.0	413167.0	413167.0	413167.0
mean	75.09019839435386	3.409432505500197	7.532377948868133	45.32000135538414	1985.1628058388012
std	1664.3864829408208	1.9846876835416725	5.763012094719905	14.328042124871322	4776.2643908703785
min	0.0	1.0	0.0	18.0	2.0
25%	0.0	2.0	3.0	34.0	67.0
50%	0.0	3.0	7.0	44.0	287.0
75%	0.0	4.0	12.0	54.0	1410.0
max	306559.0	12.0	100.0	99.0	27000.0

Slika 7.1: Numerična analiza numeričnih spremenljivk.



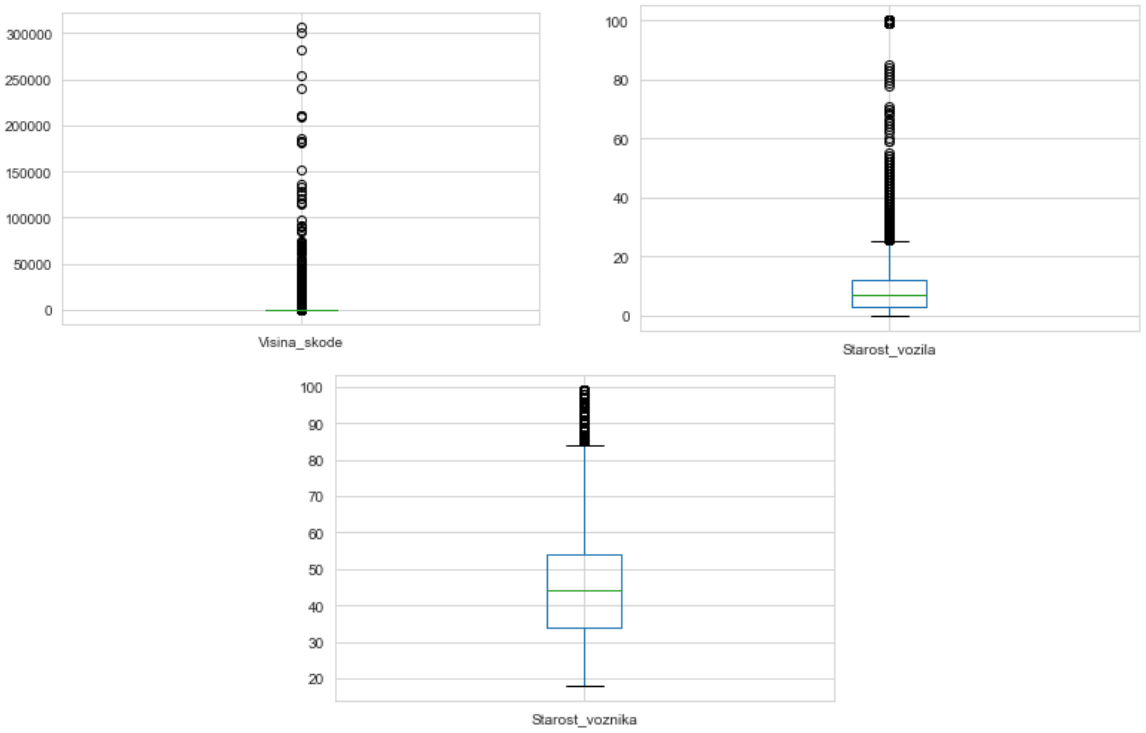
Slika 7.2: Histogrami za numerične spremenljivke.



Slika 7.3: Stolpični grafi za kategorične spremenljivke.

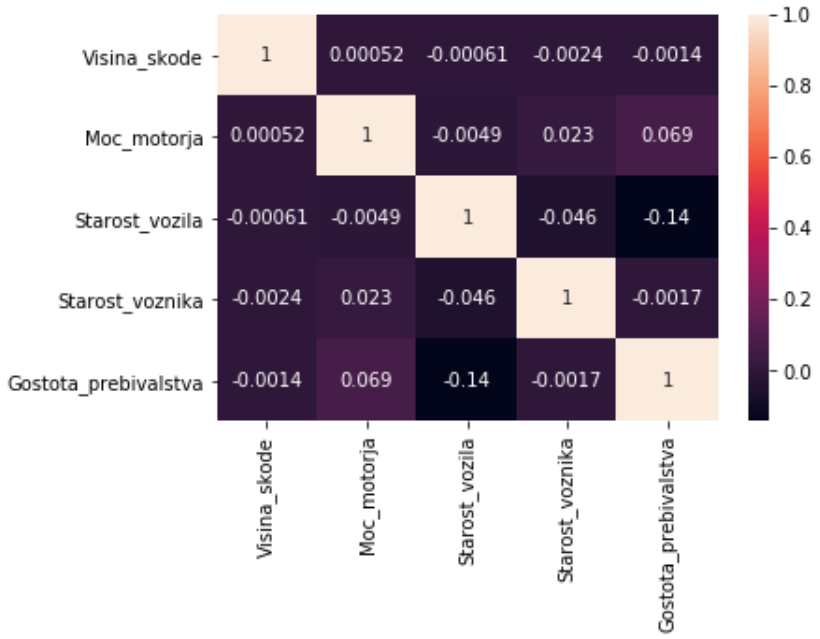
Osamelci oziroma izstopajoče vrednosti imajo lahko prevelik vpliv na naš model, kar bi pomenilo slabše napovedi, zato smo jih združili v najbližjo skupino. Osamelce smo pri posamezni numerični spremenljivki določili s pomočjo grafa imenovanega škatla z brki.

Na sliki 7.4 vidimo, da osamelce pri spremenljivki *Starost_vozila* predstavljajo vozila stara 30 let ali več, zato bomo le-te združili v skupino vozil starih 30 let. Vidimo tudi, da je število ljudi, starih 80 let ali več, zanemarljivo, zato jih bomo združili v skupino ljudi starih 80 let. Čeprav vidimo, da izstopajo velike škode, jih ne bomo združili v skupino, ker te precej vplivajo na povprečno višino škode.



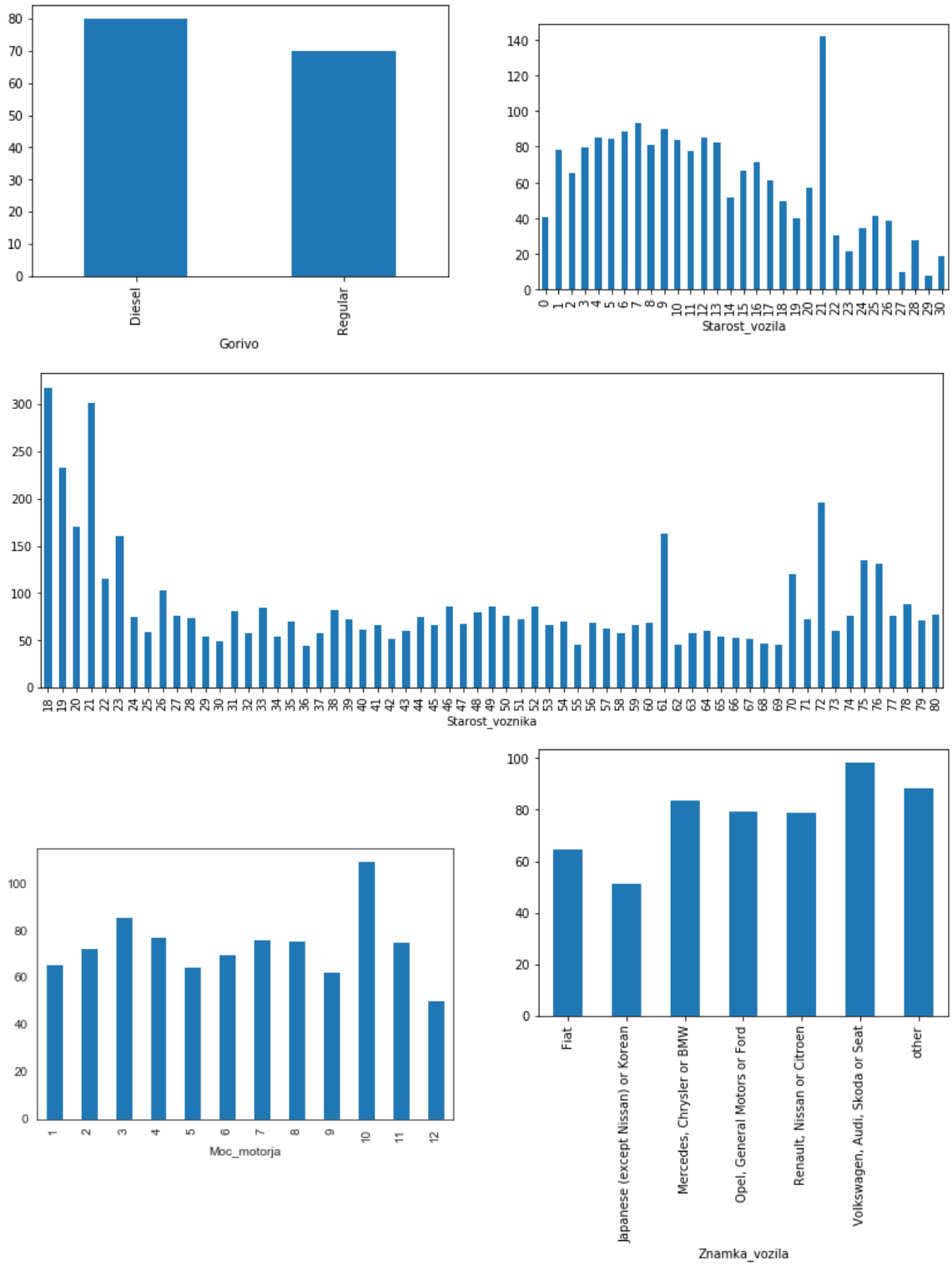
Slika 7.4: Škatle z brki za numerične spremenljivke.

Iz korelacijske matrike, ki je predstavljena na sliki 7.5, lahko sklepamo, da nastopajoče spremenljivke v matriki med sabo niso korelirane, kar pomeni, da med njimi ni linearne povezave, lahko pa seveda obstaja nelinearna povezava.



Slika 7.5: Korelacijska matrika.

Glede na nekatere vhodne spremenljivke smo proučili tudi ciljno spremenljivko *Visina_skode*. Na sliki 7.6 vidimo, da imajo vozila na bencin in dizel zelo podobno višino škode, kar pomeni, da gorivo nima pomembnega vpliva na našo ciljno spremenljivko. Mlajši vozniki imajo najvišje škode, kar je pričakovano, saj imajo le-ti najmanj izkušenj in so zato najbolj rizični. Tudi pri zelo starih voznikih je višina škode večja kot sicer, saj starejši ljudje niso več v najboljši telesni in umski pripravljenosti za vožnjo ter zato predstavljajo večjo nevarnost na cesti. Pri vozilih pa imajo najvišje škode vozila srednjih let, ki verjetno niso več optimalno pripravljena za vožnjo. Tudi vozila z močnejšimi motorji imajo višje škode, kot tista s slabšimi motorji, kar je tudi smiselno, saj je verjetnost nesreče z močnejšim motorjem večja. Če pa opazujemo višino škode glede na znamko vozila, lahko opazimo, da imajo "dražje" znamke vozil, kot so Volkswagen, Audi, Škoda, Opel in Mercedes, višje škode kot "cenejše" znamke, kot so na primer Fiat, Kia ipd.



Slika 7.6: Povprečna višina škode glede na nekatere spremenljivke.

7.2.4 Povzetek po obdelavi podatkov

Naša ciljna spremenljivka, ki smo jo želeli napovedati, je *Visina_skode*. Vhodne numerične spremenljivke so *Starost_vozila*, *Starost_voznika*, *Gostota_prebivalstva* in *Moc_motorja*. Slednjo smo za potrebe modelov pretvorili iz kategorične spremenljivke v numerično. Vhodne kategorične spremenljivke pa so *Znamka_vozila*, *Regija* in *Gorivo*, ki smo jih za uspešno izvedbo strojnega učenja pretvorili v binarne spremenljivke. Tako smo imeli po pretvorbi skupaj na voljo 23 vhodnih spremenljivk.

Pri podatkih smo osamelce združili v najbližjo skupino. Tako smo vozila, stara 30 let ali več, združili v skupino vozil starih 30 let. Tudi voznike, stare 80 let ali več, smo združili v skupino voznikov starih 80 let.

Podatke smo naključno delili na učne in testne, kar pomeni da je bila posamezna polica naključno dodeljena učni ali testni množici, pri tem je bila verjetnost dodelitve v učno množico enaka 0.80.

Poglavje 8

Kalibracija modelov

V tem poglavju smo za posamezni model določili najustreznejše parametre. Za izbiro najboljšega modela smo si pomagali s povprečno kvadratno napako oziroma vrednostjo MSE in 5-kratnim prečnim preverjanjem.

8.1 Posplošeni linearni model

Kot smo povedali v poglavju 3.4, se za modeliranje višine škode običajno uporablja gama porazdelitev, saj so velike škode redke, ničelne ali majhne škode pa pogoste. Za povezovalno funkcijo, ki jo potrebujemo pri posplošenem linearnem modelu, pa smo izbrali logaritemsko funkcijo.

Primerjali smo več modelov, ki vsebujejo različne vhodne spremenljivke. V tabeli 8.1 sta predstavljeni oznaka modela in uporabljene vhodne spremenljivke.

Model	Vhodne spremenljivke
glm_0	const
glm_1	const, Starost_voznika
glm_2	const, Starost_voznika, Starost_vozila
glm_3	const, Starost_voznika, Starost_vozila, Znamka_vozila
glm_4	const, Starost_voznika, Starost_vozila, Znamka_vozila, Moc_motorja
glm_5	const, Starost_voznika, Starost_vozila, Znamka_vozila, Moc_motorja, Regija
glm_6	const, Starost_voznika, Starost_vozila, Znamka_vozila, Moc_motorja, Regija, Gostota_prebivalstva
glm_7	const, Starost_voznika, Starost_vozila, Znamka_vozila, Moc_motorja, Regija, Gostota_prebivalstva, Gorivo

Tabela 8.1: Posplošeni linearni modeli.

Rezultate lahko vidimo v tabeli 8.2.

Model	MSE
glm	2770250.28
glm_1	2770269.08
glm_2	2770300.10
glm_3	2770133.16
glm_4	2770149.53
glm_5	2770064.00
glm_6	2770070.14
glm_7	2770070.98

Tabela 8.2: Rezultati posplošenih linearnih modelov.

Opazimo, da je najboljši model glm_5, ki vsebuje spremenljivke Starost_voznika, Starost_vozila, Znamka_vozila, Moc_motorja in Regija. Čeprav je ta model najboljši, bomo za primerjavo z naključnimi gozdovi in nevronskimi mrežami uporabili model glm_7, ki vsebuje vse spremenljivke in je le malo slabši od modela glm_5.

8.2 Naključni gozdovi

Učinkovitost naključnega gozda je odvisna od izbire njegovih parametrov. Najbolj občutljivi parametri so naslednji [12]:

- Število dreves.
  Veliko število dreves poveča stabilnost napovedi, vendar se hkrati poveča tudi čas računanja.

- Velikost učne množice.
  Običajno uporabimo 66% podatkov, vendar je smiselno testirati še druge vrednosti. V našem primeru smo za učenje uporabili 80% podatkov.

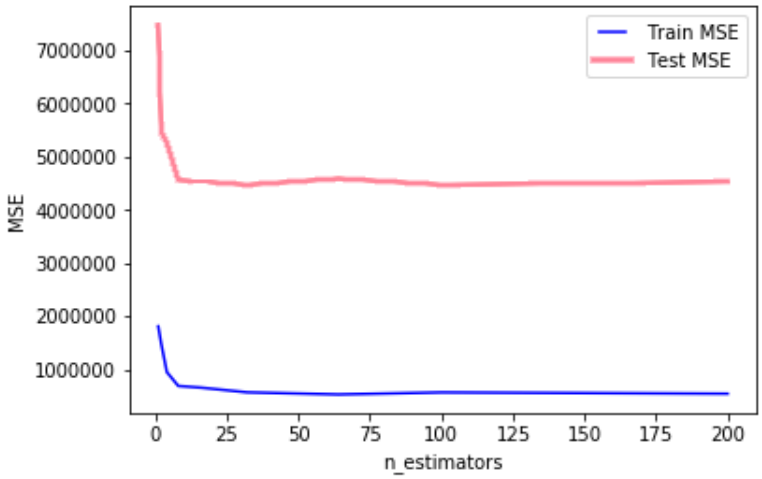
- Število spremenljivk m, med katerimi iščemo najboljšo delitev.
  Recimo, da imamo na voljo p vhodnih spremenljivk. Če izberemo $m = p$, bodo drevesa med sabo zelo podobna. Če izberemo $m = 1$, bodo drevesa naključna. Običajno vzamemo $m \approx \sqrt{p}$, kar smo uporabili tudi za naš primer.

- Največja globina dreves.
  Želimo zgraditi drevesa, ki se ne prilegajo preveč učnim podatkom in razumejo pomembnost relacij med podatki.

Da bi določili najboljše parametre, lahko testiramo vse možne kombinacije parametrov in izberemo tisto, ki proizvede najmanjšo napako. Vendar je to, zaradi prevelike računske zahtevnosti, običajno neizvedljivo. Zato se uporabi pristop, ki ga imenujemo naključno mrežno iskanje parametrov. Na začetku med vsemi kombinacijami parametrov naključno izberemo določeno število kombinacij, ki jih nato testiramo. Glede na dobljene rezultate lahko nato izvedemo podrobnejše iskanje najboljših parametrov v okolici najboljše kombinacije ter tako dobimo lokalno najboljšo kombinacijo parametrov, ki pa ni nujno globalno najboljša.

8.2.1 Število dreves

Predstavlja število dreves v gozdu. Večje število dreves omogoča boljše učenje na podatkih, vendar se lahko z dodajanjem dreves, učni proces zelo upočasni.

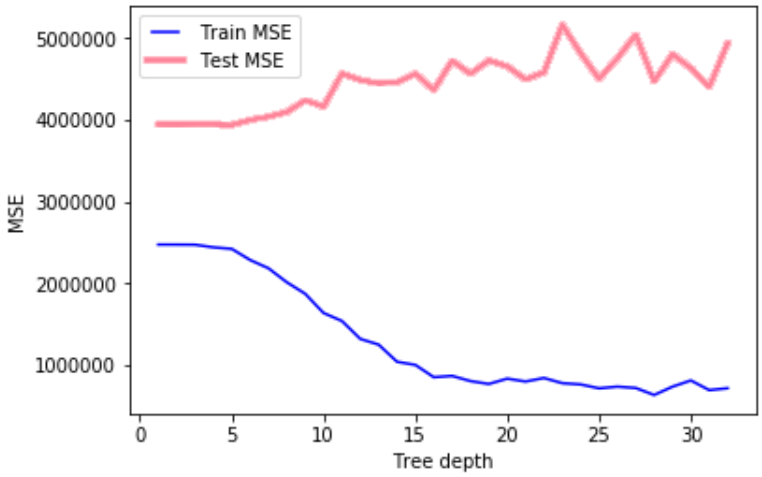


Slika 8.1: Število dreves.

Na sliki 8.1 vidimo, da se lahko ustavimo pri 60 drevesih, saj se z večjim številom dreves natančnost ne izboljša.

8.2.2 Največja globina dreves

Predstavlja največjo dovoljeno globino posameznega drevesa v gozdu. Globlje drevo ima več delitev in tako zajame več informacij o podatkih, vendar lahko pride do prevelikega prilagajanja učnim podatkom in posledično slabšega prilagajanja testnim podatkom.



Slika 8.2: Globina dreves.

Na sliki 8.2 vidimo, da se globja drevesa preveč prilagajajo učnim podatkom in slabo prilagajajo testnim podatkom, zato se je smiselno ustaviti pri globini 10.

8.2.3 Naključno mrežno iskanje parametrov

Ko smo postopek naključnega mrežnega iskanja parametrov izvedli na naših podatkih, smo dobili rezultate, ki so predstavljeni v tabeli 8.3.

Rang	MSE	n_estimators	max_depth	min_samples_leaf	min_samples_split
1	2769237.95	18	4	5	150
2	2769239.47	18	4	5	100
3	2769240.55	18	4	7	150
4	2769257.10	19	4	5	150
5	2769259.40	19	4	6	150
6	2769273.13	19	4	4	50

Tabela 8.3: Najboljših 6 naključnih gozdov, dobljenih s postopkom naključnega mrežnega iskanja parametrov.

Razložimo še pomen zadnjih štirih stolpcev. Stolpec *n_estimators* predstavlja število dreves v gozdu, *max_depth* predstavlja največjo dovoljeno globino dreves, *min_samples_leaf* predstavlja najmanjše število podatkov potrebnih za izdelavo lista ter *min_samples_split* predstavlja najmanjše število podatkov potrebnih za delitev.

Opazimo, da ima najučinkovitejši naključni gozd 18 dreves, največjo globino dreves 4 nivoje, najmanjše število podatkov potrebnih za izdelavo lista 5 in najmanjše število podatkov potrebnih za delitev 150.

8.2.4 Najboljši naključni gozd

Če povzamemo, ima najboljši naključni gozd za naše podatke naslednje parametre:

- Število dreves: 18.
- Velikost učne množice: 80%.
- Število obravnavanih spremenljivk pri vsaki delitvi: $\sqrt{23}$.
- Največja globina dreves: 4.
- Najmanjše število podatkov za izdelavo lista: 5.
- Najmanjše število podatkov, potrebnih za delitev: 150.

8.3 Nevronske mreže

Tudi učinkovitost nevronske mreže je odvisna od izbire njenih parametrov. Najbolj občutljivi parametri so naslednji:

- Aktivacijska funkcija.
- Število nivojev in nevronov v posameznem nivoju.
- Stopnja učenja.

Tako kot pri naključnih gozdovih smo najboljše parametre za nevronsko mrežo poiskali z metodo naključnega mrežnega iskanja parametrov.

Za učinkovitejše učenje nevronske mreže smo podatke tudi normalizirali s pomočjo formule

$$x_{novi} = \frac{x_{stari} - x_{min}}{x_{max} - x_{min}},$$

kjer sta x_{min} in x_{max} najmanjši oziroma največji element posamezne spremenljivke. S tem smo dosegli, da vse vrednosti padejo med 0 in 1. Za obravnavo napovedi pa smo uporabili inverzno transformacijo.

8.3.1 Aktivacijska funkcija

Aktivacijska funkcija transformira uteženo vsoto vhodnih signalov v izhodni signal, katerega pošljemo naprej po mreži. V našem primeru smo za učenje nevronske mreže izbrali logistično sigmoidno funkcijo, ki je podana s predpisom

$$f(x) = \frac{1}{1 + e^{-x}}.$$

To funkcijo smo izbrali, ker ne pričakujemo negativnih škod. Razpon funkcije je interval $(0, 1)$, ki je primeren za pozitivnost in je motivacija za normalizacijo podatkov.

8.3.2 Število nivojev in nevronov v posameznem nivoju

Število nivojev in nevronov v posameznem nivoju predstavlja število skritih nivojev in število nevronov v njih. Obstaja neskončno mnogo kombinacij. V našem primeru smo testirali par možnosti in dobili rezultate, ki jih lahko vidimo v tabeli 8.4.

Število skritih nevronov v nivoju	MSE
(12)	2770170.74
(23)	2769982.48
(46)	2770332.04
(16, 8)	2770476.18
(23, 12)	2770266.78
(50, 25)	2770837.82
(70, 35)	2770381.90

Tabela 8.4: Skriti nevroni.

Opazimo lahko, da v našem primeru dobimo boljše rezultate z nevronsko mrežo z enim skritim nivojem, kot pa z uporabo dveh skritih nivojev. Vidimo, da ima najboljši rezultat nevronska mreža z enim skritim nivojem, ki vsebuje 23 nevronov. Tudi nevronska mreža z enim skritim nivojem, ki vsebuje 12 nevronov, je pokazala dobre rezultate. Pri nevronskih mrežah z dvema skritima nivojema je bila najuspešnejša mreža, ki ima 23 in 12 nevronov.

8.3.3 Stopnja učenja

Stopnja učenja določa, kako hitro ali počasi se nevronska mreža uči. S tem parametrom nadzorujemo spremembo uteži glede na ocenjeno napako. Manjša kot je ta vrednost, počasneje se mreža uči in posledično se čas učenja podaljša.

Stopnja učenja	MSE
0.01	2770285.91
0.05	2771369.67
0.001	2770179.53
0.005	2770311.33
0.0001	2770074.80
0.0005	2770166.62

Tabela 8.5: Stopnja učenja.

Iz tabele 8.5 lahko razberemo, da je za naše podatke najuspešnejša stopnja učenja 0.0001, vendar imata 0.0005 in 0.001 tudi dober rezultat, zato bomo v okolici teh števil iskali najustreznejšo vrednost.

Pri nevronskih mrežah še lahko določimo parameter *Velikost skupine*, ki pove, koliko podatkov naenkrat pošljemo čez mrežo oziroma koliko podatkov naenkrat uporabimo za učenje mreže.

8.3.4 Naključno mrežno iskanje parametrov

Ko smo naključno mrežno iskanje parametrov izvedli na naših podatkih, smo dobili rezultate, ki so predstavljeni v tabeli 8.6.

Rang	MSE	Stopnja učenja	Število skritih nevronov	Velikost skupine
1	2769979.80	0.001	(23)	100
2	2770008.80	0.001	(70, 35)	200
3	2770074.41	0.001	(23)	50
4	2770077.57	0.0001	(12)	100
5	2770113.28	0.001	(12)	200
6	2770153.08	0.0005	(50, 25)	50

Tabela 8.6: Najboljših 6 nevronskih mrež, dobljenih s postopkom naključnega mrežnega iskanja parametrov.

Glede na vrednost MSE opazimo, da ima najučinkovitejša nevronska mreža en skriti nivo s 23-imi nevroni, stopnjo učenja enako 0.001 ter velikost skupine 100 podatkov.

8.3.5 Najboljša nevronska mreža

Če povzamemo, ima najboljša nevronska mreža za naše podatke naslednje parametre:

- Aktivacijska funkcija: logistična sigmoidna funkcija, $f(x) = \frac{1}{1+e^{-x}}$.
- Stopnja učenja: 0.001.
- Število skritih nivojev in nevronov: 1 skriti nivo s 23-imi nevroni.
- Velikost skupine: 100.

Poglavje 9

Primerjava modelov

9.1 Primerjava

V tem poglavju smo primerjali uspešnost posameznih modelov. Za ocenjevanje uspešnosti modelov smo uporabili povprečno kvadratno napako oziroma vrednost MSE in 5-kratno prečno preverjanje.

Primerjali smo naslednje modele:

- otm.
  Osnovni testni model, pri katerem ima vsaka polica višino škode enako povprečni višini škode, ki je izračunana na učnih podatkih.
- glm.
  Posplošeni linearni model, ki je predstavljen v podpoglavju 8.1 in vsebuje vse spremenljivke.
- rf.
  Naključni gozd, ki je predstavljen v podpoglavju 8.2.4.
- nn.
  Nevronska mreža, ki je predstavljena v podpoglavju 8.3.5.

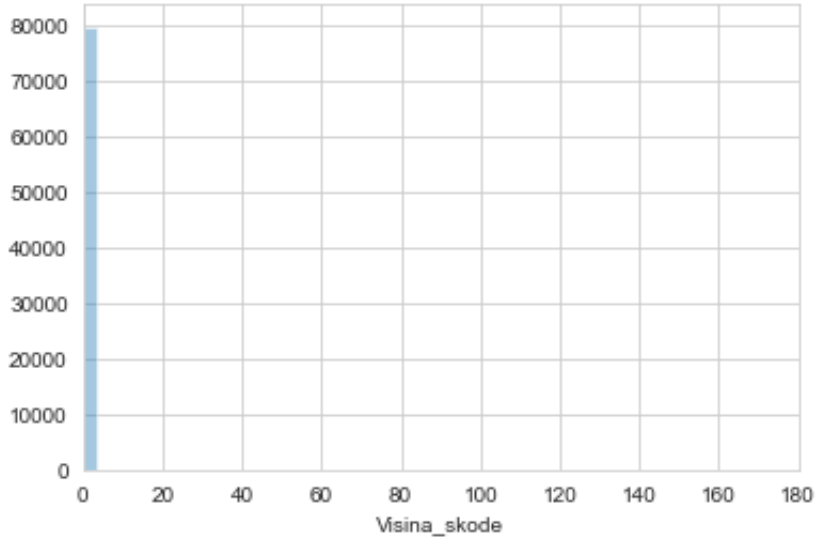
Vrednosti MSE pri posameznem modelu lahko vidimo v tabeli 9.1.

Model	MSE
otm	2770250.28
glm	2770070.98
rf	2769234.25
nn	2769979.80

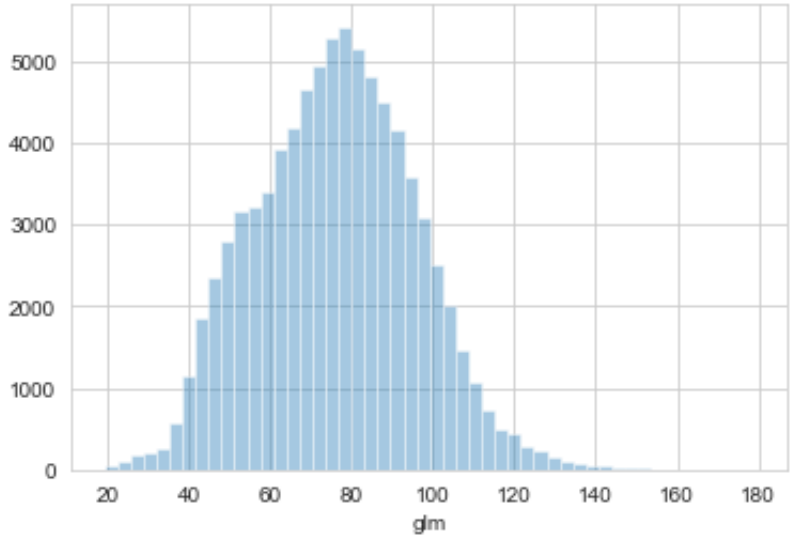
Tabela 9.1: Primerjava modelov.

Opazimo lahko, da so vsi modeli imeli boljši rezultat kot osnovni testni model. Glede natančnosti se je najbolje odrezal naključni gozd, ki je imel najmanjšo napako. V primerjavi z osnovnim testnim modelom se je natančnost napovedi pri naključnem gozdu izboljšala za 0.018%. Malo slabše se je odrezala nevronska mreža, pri kateri se je natančnost napovedi izboljšala za 0.010%. Pri posplošenem linearnem modelu pa se je natančnost napovedi izboljšala za samo 0.006%.

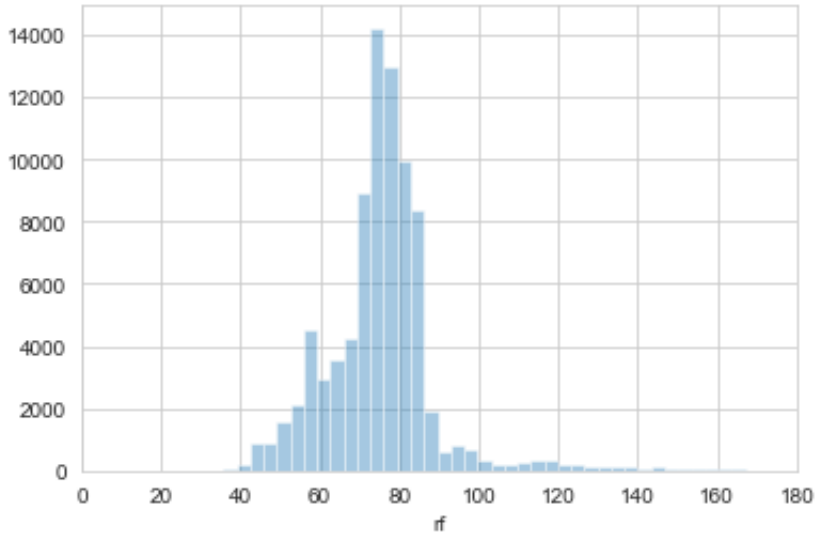
Poglejmo si še porazdelitev napovedi za posamezni model. V ta namen smo naključno izbrali 80% podatkov za našo učno množico, preostalih 20% pa za našo testno množico. Grafično si bomo porazdelitve ogledali za škode, ki so manjše od vrednosti 180, saj je večina napovedi med 0 in 180.



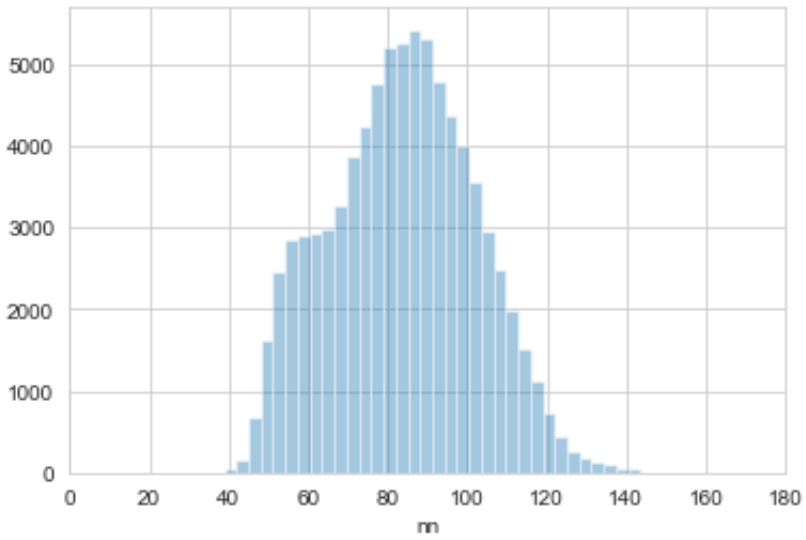
Slika 9.1: Porazdelitev dejanske višine škod v testni množici.



Slika 9.2: Porazdelitev napovedi pri posplošenem linearnem modelu.



Slika 9.3: Porazdelitev napovedi pri naključnem gozdu.



Slika 9.4: Porazdelitev napovedi pri nevronski mreži.

Vidimo lahko, da se pri vseh modelih porazdelitev napovedi precej razlikuje od dejanske porazdelitve. Pri vseh modelih je večina napovedi okoli vrednosti 80. Opazimo lahko, da imata GLM in nevronska mreža podobno porazdelitev, medtem ko je porazdelitev pri naključnem gozdu ožja in bolj stisnjena ter zato najbolj podobna dejanski porazdelitvi škod.

Pri numerični analizi porazdelitev pa smo upoštevali vse škode.

	glm	rf	nn	original
count	82634.0	82634.0	82634.0	82634.0
mean	76.00068154979627	75.95662555001506	83.57563332779858	71.28900936660455
std	19.62113029993886	20.794315001042765	18.35256680900093	1237.337421226672
min	19.165652453325734	35.67654371150552	38.84247339454435	0.0
25%	61.889122761887975	69.1180465180675	70.29500409166656	0.0
50%	76.2404619682935	75.56129086271771	83.95750753507188	0.0
75%	89.458906104409	80.68717968760194	96.5439256305592	0.0
max	179.11905219038195	709.1847788379921	146.64757760687817	181697.0

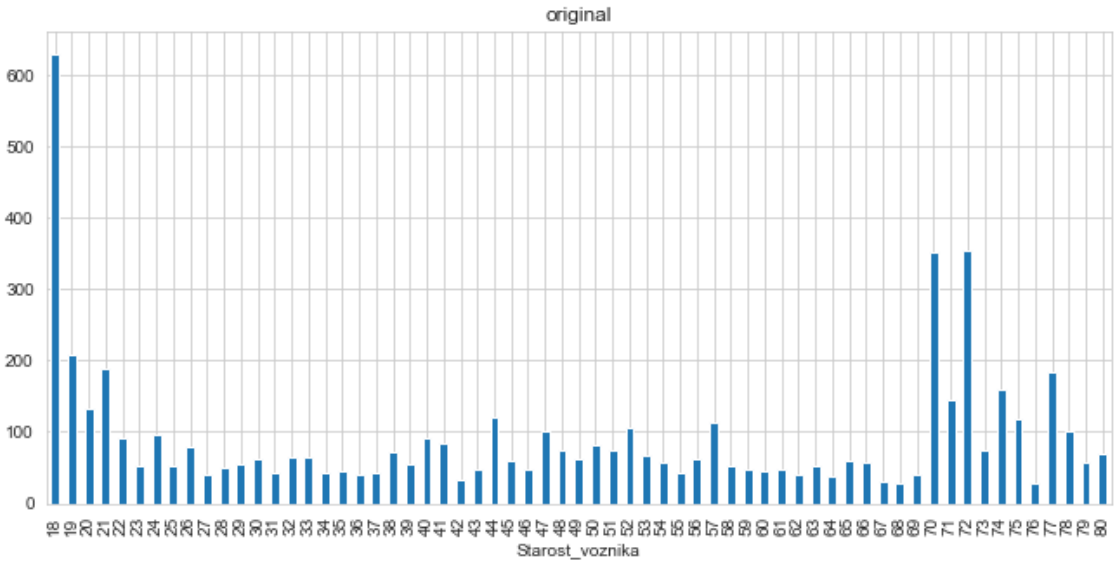
Slika 9.5: Numerični analiza porazdelitev napovedi.

Opazimo, da je povprečna napoved škode pri GLM-ju in naključnem gozdu blizu dejanski povprečni škodi, medtem ko je povprečna napoved škode pri nevronski mreži nekoliko višja. Naključni gozd in nevronska mreža, v primerjavi z GLM-jem, ne napovedujeta zelo nizkih škod, saj imata najmanjšo napoved pri vrednosti 35.68 oziroma 38.84. Pri napovedovanju

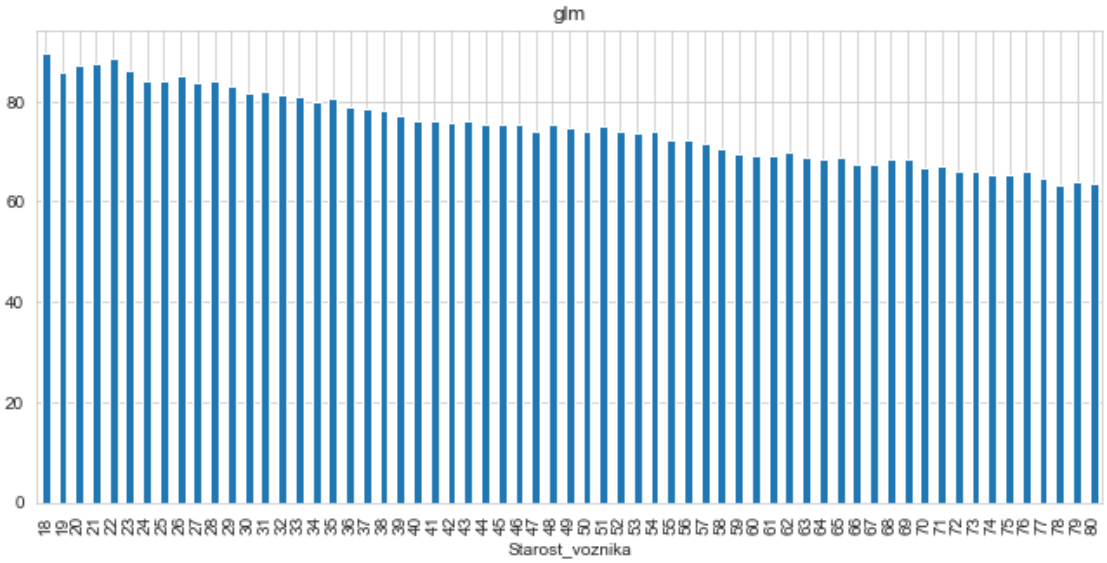
velikih škod pa imata GLM in nevronska mreža zelo nižje napovedi kot naključni gozd. Noben model pa ni napovedal dejanske najvišje škode, ki je 181697.

Po vsem tem lahko sklepamo, da se je dejanski porazdelitvi škod najbolj približal naključni gozd, ki je imel tudi najmanjšo napako napovedi. Zato je v našem primeru za napovedovanje višine škode najboljši model naključni gozd.

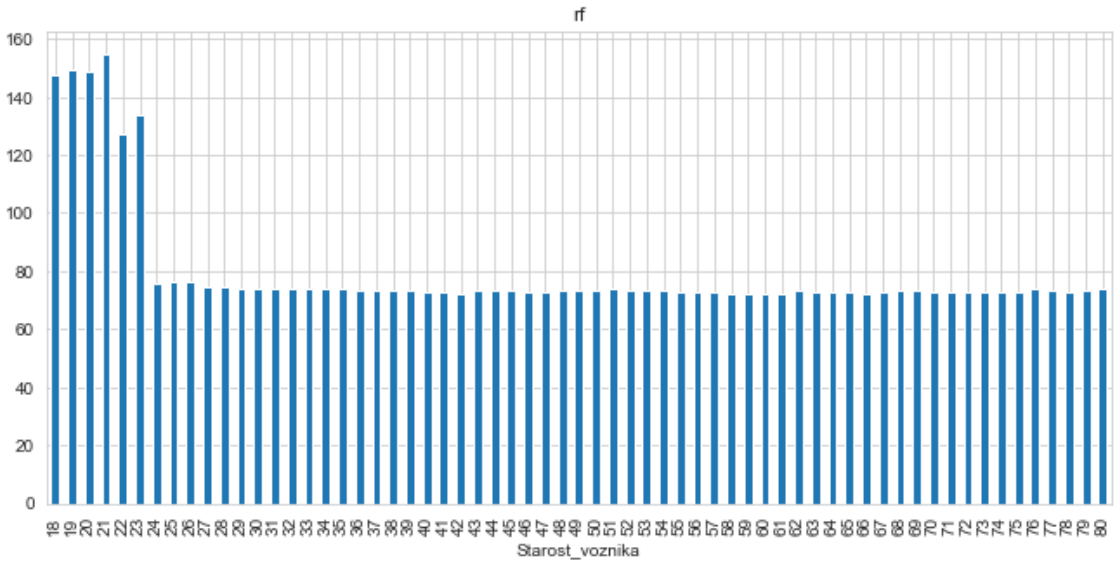
Zdaj pa še poglejmo višino škode glede na starost voznika na dejanskih podatkih in po posameznih modelih. Rezultati so prikazani na slikah 9.6, 9.7, 9.8 in 9.9. Opazimo lahko, da so vsi modeli prepoznali, da imajo mlajši vozniki višje škode, kot starejši. Pri GLM-ju in nevronski mreži višina škode skoraj linearno pada z večanjem starosti, medtem ko pri naključnem gozdu to ne velja. Po obliki so dejanskim podatkom še najbližje napovedi naključnega gozda. Noben model pa ni pokazal, da imajo ljudje, ki so starejši od 70 let, višje škode, kot ljudje srednjih let. To lahko prepoznamo kot neko pomanjkljivost vseh modelov.



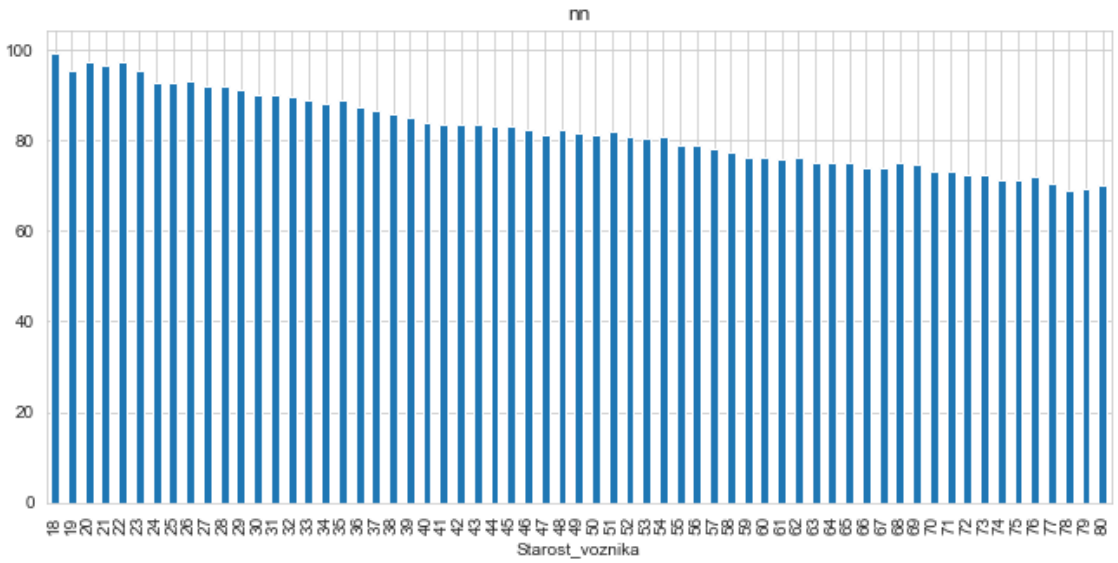
Slika 9.6: Višina škode glede na starost voznika pri dejanskih podatkih.



Slika 9.7: Višina škode glede na starost voznika pri poslošenem linearnem modelu.



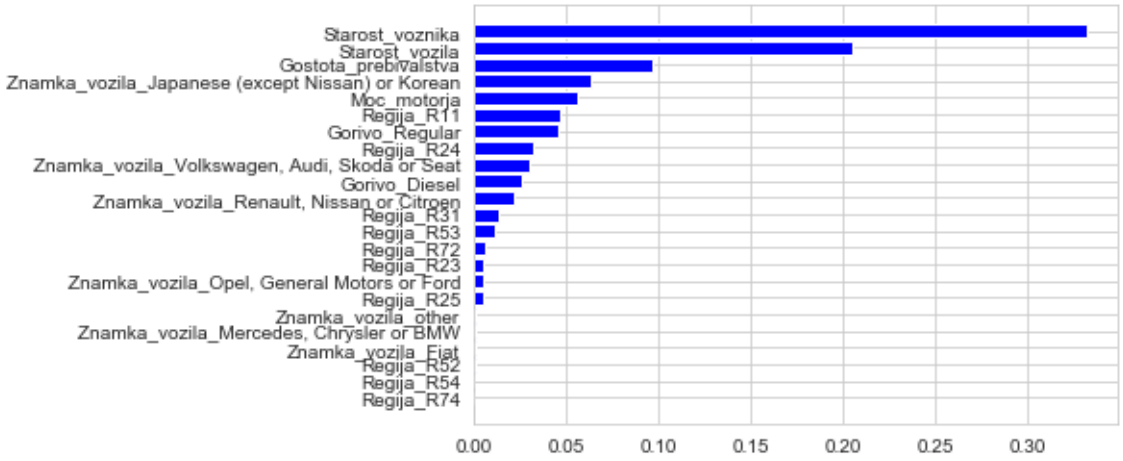
Slika 9.8: Višina škode glede na starost voznika pri naključnem gozdu.



Slika 9.9: Višina škode glede na starost voznika pri nevronski mreži.

9.2 Pomembnost spremenljivk

Kot smo omenili v poglavju 5.4.1, lahko s pomočjo naključnega gozda pridemo do spremenljivk, ki so pri postopku napovedovanja najpomembnejše. Za naš primer dobimo rezultate, ki so prikazani na sliki 9.10.



Slika 9.10: Pomembnost spremenljivk.

Vidimo lahko, da sta za napovedovanje višine škode najpomembnejši lastnosti oziroma spremenljivki *Starost_voznika* in *Starost_vozila*, kar je tudi smiselno, saj na verjetnost nesreče najbolj vplivajo izkušnje voznika in starost materiala. Tudi *Gostota_prebivalstva*,

Znamka_vozila in *Moc_motorja* imajo pomembno vlogo pri napovedovanju škode, medtem ko *Regija* in *Gorivo* nimata velikega vpliva na škodo.

9.3 Dodatek

V tabeli 9.2 so predstavljena izračunana razmerja med dejansko in napovedano vsoto vseh škod.

otm	glm	rf	nn
0.937536	0.938004	0.9385489	0.852987

Tabela 9.2: Razmerje med dejansko in napovedano vsoto vseh škod pri posameznem modelu.

Razmerje je pri vseh modelih manjše kot 1, kar iz aktuarskega vidika pomeni, da so vsi modeli pravični in primerni za določitev nevarnostne premije. Ker je to razmerje manjše kot 1, bi zavarovalnica v primeru, da bi za nevarnostno premijo določila napovedano višino škode, lahko nastale škode poplačala. S strani dobička bi se za zavarovalnico najbolje izplačala nevronska mreža, ki ima to razmerje najnižje in posledično bi zavarovalnica s tem modelom imela največji profit. Glede na podatke pa je najpravičnejši model za zavarovalnico in zavarovanca naključni gozd, ki ima to razmerje najbližje vrednosti 1 in je v tem pogledu celo boljši kot posplošeni linearni model, ki ga zavarovalnice običajno uporabljajo za izračun nevarnostne premije.

9.4 Povzetek rezultatov

Za napovedovanje višine škode smo uporabili posplošeni linearni model, naključni gozd in nevronsko mrežo. Vsi trije modeli so podali podobne napovedi in rezultate. Najbolje se je odrezal naključni gozd, ki je imel najmanjšo napako napovedi in tudi porazdelitev napovedi je bila najbližja dejanski porazdelitvi škod, vendar sta tudi nevronska mreža in posplošeni linearni modela imela zadovoljive rezultate.

Ugotovili smo tudi, da sta pri napovedovanju višine škode najpomembnejši lastnosti starost voznika in vozila, medtem ko gorivo in regija bivanja voznika nimata pomembnega vpliva.

Čeprav se noben model ni v celoti približal dejanskim vrednostim višine škod, kar tudi nismo pričakovali, smo pokazali, da lahko s pomočjo strojnega učenja pridemo do boljših rezultatov, kot s posplošenim linearnim modelom. Dobljeni rezultati so zanimivi tudi zato, ker bi lahko napovedi uporabili za pravično določitev nevarnostne premije.

Poglavje 10

Zaključek in odprta vprašanja

Prepoznavanje rizičnih in nerizičnih posameznikov je v zavarovalništvu zelo pomembno. Danes se za to uporabljajo posplošeni linearni modeli. Ti nam poleg tega omogočajo tudi izračun nevarnostne premije, ki je potrebna za kritje škod v prihodnosti. V tem magistrskem delu smo poskušali posplošeni linearni model zamenjati z modeli strojnega učenja, tj. z naključnim gozdom in nevronsko mrežo. Iz rezultatov v podpoglavju 9.1 lahko vidimo, da sta nevronska mreža in naključni gozd bolje napovedala višino škode, kot posplošeni linearni model, čeprav so bile vse napovedi podobne. Poudariti moramo, da je tako pri GLM-ju kot tudi pri naključnih gozdih in nevronskih mrežah na voljo precej izboljšav.

Med delom se nam je pojavilo veliko vprašanj, ki bi lahko bila tema prihodnjih raziskovanj. Tukaj je nekaj primerov:

- Kateri so optimalni parametri za nevronsko mrežo in naključni gozd?
- Kako bi podatke še lahko spremenili, transformirali, klasificirali ali pogrupirali?
- Ali obstaja kakšna kombinacija metod strojnega učenja, ki bi dala še boljše rezultate?
- Ali uspešnost modela ocenimo samo s pomočjo MSE-ja, ali upoštevamo tudi porazdelitev napovedi?
- Katere lastnosti upoštevati pri napovedovanju višine škode?
- Katere metode strojnega učenja bi še lahko uporabili?

Zaključimo lahko, da v bližnji prihodnosti strojno učenje še ne bo zamenjalo GLM-ja v zavarovalništvu, saj lahko GLM izračunamo hitreje kot modele strojnega učenja. Zato

zavarovalnice za napovedovanje še vedno uporabljajo GLM, vendar pri nekaterih primerih prepoznavanja vzorcev že uvajajo strojno učenje, ki ima zmožnost odkrivanja skritih vzorcev.

Posplošeni linearni model se je v zavarovalništvu razvijal kar nekaj časa, medtem ko se je strojno učenje pojavilo pred razmeroma kratkim časom, zato obstaja še veliko stvari neraziskanih. Metode strojnega učenja se razvijajo zelo hitro, tako da poleg naključnih gozdov in nevronskih mrež obstaja še veliko drugih metod, kot na primer metoda podpornih vektorjev, metoda k najbližjih sosedov in mnoge druge.

Glede na to, da smo šele na začetku razvoja strojnega učenja, verjamemo, da ima strojno učenje velik potencial in da se bo v prihodnosti uveljavilo ne samo v zavarovalništvu, ampak tudi na ostalih področjih.

Literatura

[1] J. Boncelj, *Zavarovalna ekonomika*, Obzorja, Maribor, 1983.

[2] E. Bubnič, L. Fric, Š. Ivanjko, M. Musil, G. Ristin, *Učbenik za zavarovalne zastopnike in zavarovalne posrednike*, Slovensko zavarovalno združenje, Ljubljana, 2016.

[3] M. Bijelić, *Zavarovanje in pozavarovanje*, Art agencija, Ljubljana 1998.

[4] M. Erker, *Strojno učenje : vrednotenje v zavarovalništvu*, Univerza v Ljubljani, Fakulteta za matematiko in fiziko, Ljubljana, 2018.

[5] D. Klančar, *Uporaba posplošenega linearnega modela v zavarovalništvu*, Univerza v Ljubljani, Fakulteta za matematiko in fiziko, Ljubljana, 2015.

[6] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning with applications in R*, Springer, London, UK, 2013.

[7] B. Lantz, *Machine Learning with R*, Second Edition, Packt Publishing Ltd, Birmingham, UK, 2015.

[8] S. Jamal, S. Canto, R. Fernwood, C. Giancaterino, M. Hiabu, L. Invernizzi, T. Korzhynska, Z. Martin, H. Shen, *Machine Learning and Traditional Methods Synergy in Non-Life Reserving*, ASTIN working party report, 2018.

[9] S. Čoh, *Test dobičkonosnosti pri razvoju mešanega življenjskega zavarovanja*, Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Maribor, 2017.

[10] H.I. Witten, E. Frank, A.M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3. edition, Morgan Kaufmann Publishers, 2011.

[11] B. Harej, R. Gächter, S. Jamal, *Individual Claim Development with Machine Learning*, ASTIN working party report, 2017.

[12] S. Jamal, S. Canto, R. Fernwood, C. Giancaterino, M. Hiabu, L. Invernizzi, T. Korzhynska, Z. Martin, H. Shen, *Machine Learning and Traditional Methods Synergy in Non-Life Reserving*, ASTIN working party report, 2018.