Strojno učenje u nastavi informatike u prirodoslovnim gimnazijama

Ljubičić, Ivan

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivan Ljubičić

STROJNO UČENJE U NASTAVI INFORMATIKE U PRIRODOSLOVNIM GIMNAZIJAMA

Diplomski rad

Voditelj rada:
dr.sc. Tomislav Šmuc

Zagreb, Rujan, 2018.

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

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

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj rad posvećujem, u proporcionalnom odnosu, svima koji su me oblikovali u osobu koja danas jesam.

Sadržaj

Uvod

U današnje doba interneta i najave interneta stvari, jedan smjer istraživanja postaje sve popularniji i sveprisutniji. Riječ je o umjetnoj inteligenciji i njenoj grani, strojnom učenju. Otkada je internet započeo svoje strmoglavo širenje, strojno učenje je zaživilo u gotovo svim ljudskim djelatnostima. U ovom radu nastojat ću naglasiti važnost strojnog učenja u današnjem životu, definirati osnovne pojmove strojnog učenja i obraditi nekoliko algoritama strojnog učenja s primjerima. Osnovni cilj ovog rada je dati smjernice za uvođenje strojnog učenja u školski sustav, specifično u prirodoslovne gimnazije.

Na samom početku definirat ću osnovnu podjelu strojnog učenja. Objasnit ću razlike svih tih dijelova kao i navesti računalne algoritme vezane uz iste. Nakon toga ću se posvetit detaljnoj razradi nekoliko algoritama strojnog učenja i njihovom uvođenju u nastavnom okruženju. Nastojat ću definirati potrebna predznanja i obrazovne ciljeve vezane uz svaki algoritam. Pred kraj rada pokušat ću dati širu sliku primjene strojnog učenja u svakodnevnom životu.

Ovaj rad je pisan u smislu nadopune nastave informatike u prirodoslovnim gimnazijama ili općenito bilo gdje, ako to predznanje učenika dopušta. Većina obrađenog sadržaja namijenjena je naprednijim učenicima dok se primjeri primjene strojnog učenja mogu obrađivati kod teme kao što su sigurnost na internetu.

Poglavlje 1

Strojno učenje

1.1 Učiti kako učiti

U današnjem obrazovanju javlja se jedna veoma važna tema kada se govori o međupredmetnim temama. Ta tema je *Učiti kako učiti*. Danas se radna okolina mijenja velikom brzinom i zahtijevaju od radnika mogućnost brze prilagodbe i stjecanja novih vještina dok je prije bilo moguće s jednom naučenom vještinom osigurati svoj opstanak do kraja radnog vijeka. [2] Svrha je uvođenja međupredmetne teme *Učiti kako učiti* omogućiti učenicima da razviju znanja i vještine upravljanja svojim učenjem i primjene odgovarajućih strategija u različitim situacijama učenja u formalnom, neformalnom i informalnom okruženju. Ta će im kompetencija omogućiti da usvoje znanja i vještine koje će moći uspješno primjenjivati u kasnijem osobnom i profesionalnom razvoju u kontekstu cjeloživotnog učenja.

Moje je mišljenje i iskustvo da se najbolje uči dok se druge podučava. Kroz proces podučavanja mi dobivamo povratne informacije koje nam dopuštaju dublji uvid u samu prirodu učenja. Proces podučavanja od nas zahtjeva da mijenjamo strategiju i način podučavanja kako bi dobili bolji rezultat. Jedan od načina na koji se mogu steći spoznaje i vještine je kroz osvještavanje. Djeca od najranije dobi pokušavaju razne strategije učenja kako bi dobili što bolje rezultate. Kroz metodu pokušaja i promašaja procjenjuju koja metoda učenja njima najbolje odgovara. Sve se to, naravno, događa na potpuno intuitivnoj razini, tek u trenutku kad oni osvijeste svoje strategije učenja mogu dobiti dublji uvid u stečeno znanje te dolazi do onog „Ahaaa“ trenutka. Rijetko koji osjećaj je ljepši od onog kada se napokon shvati razlog zašto smo sve godine radili nešto bez da točno znamo zašto. U takvim trenucima imamo to zadovoljstvo spoznaje da nas naša intuicija dobro vodi i da želimo ići još više istražiti to područje. Barem je kod mene bilo i ostalo tako. I točno tu ulazi važnost strojnog učenja kao predmeta.

Strojno učenje se bavi podučavanjem računala, a kao što to obično bude, znanost oponaša prirodu. Algoritmi koji se koriste za učenje računala su vezani, na jednoj ap-

straktnoj konceptualnoj razini, s načinom na koji ljudi uče. Ideja je ovog rada da kroz podučavanje strojnog učenja poveže njegove koncepte s međupredmetnom temom *Učiti kako učiti*. Strojno učenje dijeli se na tri koncepta: nadzirano učenje (supervised learning), nenadzirano učenje (unsupervised learning) i podržano učenje (reinforced learning). U daljnjem tekstu će se obraditi ti koncepti te navesti primjeri algoritama koji se koriste kod svakog od njih.

1.2 Nadzirano učenje

Došao je taj dan. Mate je na nastavi matematike počeo učiti tablicu množenja. Pred njim se sada nalazi veliki izazov, da nauči tablicu množenja napamet. Mate ide doma i uči tablicu, svoje rezultate provjerava i zadovoljan je sa svojim napretkom. Dolazi roditelju kojeg zamoli da ga ispita. Ispitivanje krene i Mate daje svoje odgovore. Ovisno o odgovorima Mate od svog roditelja dobiva povratnu informaciju o točnosti danog rješenja. Ubrzo shvaća da nije toliko znao koliko je mislio, ali roditelj je uporan u ispitivanju pa Mate postane nešto bolji u tablici množenja. Dođe dan kad će nastavnik pitati tablicu množenja i cijeli razred priča o tome. Svi jedni druge ispituju. Kroz razgovor sa školskim kolegama Mate još malo uči prije ispitivanja dok mu prijatelji govore dali je točno ili netočno odgovorio. Konačno počinje sat matematike i nastavnik krene s ispitivanjem tablice množenja. Ovaj put Mate već zna sve odgovore, jer je u više navrata učio i zapamtio odgovore te dobiva odličnu ocjenu.

Ova kratka priča opisuje primjenu nadziranog učenja u svakodnevnom životu. Nadzirano učenje se svodi na to da na temelju danih podataka, kroz povratne informacije, naučimo određeno ponašanje. U gornjem slučaju podaci koje imamo su: zadatak, Matino rješenje i informaciju dali je rješenje točno ili netočno, a ponašanje koje želimo naučiti je točno rješavanje zadataka iz tablice množenja. Ovakav problem spada pod problem klasifikacije. Problem klasifikacije je jedno od dva glavna problema koja rješava nadzirano učenje, drugi je regresija. U daljnjem tekstu će se obraditi ta dva problema i navesti algoritme nadziranog učenja koji rješavaju navedene probleme.

Klasifikacija

[15] Kod ove vrste problema računalni program treba odredi kojoj od k, $k \in \mathbb{N}$, klasa pripada dani unos. Kako bi algoritam učenja to postigao treba vratiti funkciju klasifikacije:

$$f : \mathbb{R}^n \longrightarrow \{1, 2, \ldots, k\}, k \in \mathbb{N}.$$

Jedna od najpoznatijih skupova podataka na kojom se ilustrira klasifikacijski problem je skup podataka cvijeta iris, hrvatskog naziva perunika. [13]Ovaj skup podataka se prvi

put spominje u radu autora R.A. Fisher naslova: The use of multiple measurements in taxonomic problems. Skup podataka se sastoji od pet parametara: dužine latica, širine latica, dužine čašičnog lista, širine čašičnog lista i vrsta cvjeta Iris koji se mogu vidjeti na slici 1.1. Zadnji parametar se dijeli na tri kategorije: Iris setosa, Iris versicolor i Iris virginica. Prva četiri parametra predstavljaju ulazne vrijednosti funkcije klasifikacije dok zadnji parametar, imena cvjetova, predstavljaju klase. Algoritam učenja bi nam, u ovom slučaju, trebao vratiti funkciju:

$$f : \mathbb{R}^4 \longrightarrow \{Iris\ setosa, Iris\ versicolor, Iris\ virginica\}.$$

Slika 1.1: Cvijet Iris Versicolor s označenim dijelovima i mjerama.

Problem kod ovakvog zapisa funkcije je taj što kodomena nije podskup skupa prirodnih brojeva. To lako riješimo tako da svaki element kodomene zamijenimo s različitim prirodnim brojevima.

Dok je vrlo jednostavno odrediti što funkcija klasifikacije treba raditi, pravi izazov je odrediti pravilo preslikavanja. U tu svrhu se koriste razni algoritmi učenja, a neki od njih mogu biti: linearna regresija, najbliži susjedi, naivni Bayes, stabla odlučivanja, neuralne mreže, metoda potpunih vektora (Support Vector Machines ili SVM) i dr. Neke od navedenih algoritama budem, u daljnem toku rada, detaljno obradio kao i neke moguće načine uvođenja istih u nastavi.

Regresija

Kod ove vrste problema [15]računalni program treba predvidjeti realnu vrijednost za neki dani unos. Kako bi algoritam učenja to postigao treba vratiti funkciju regresije:

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}.$$

Glavna razlika klasifikacije i regresije leži u tome što je klasifikacija diskretan matematički problem dok regresija nije. Uzmimo primjer procjenu težine čovjeka. Kada bi smo nekome dali zadatak da grupu ljudi poreda po težini od najlakšeg do najtežeg, bez da mu damo informaciju o težini svake osobe, ta osoba bi bila prisiljena pogađati. Ali kako? S velikom vjerojatnošću će svoju procjenu bazirati na visini i stasu. Dok su visoki i krupni ljudi sigurno teži, niski i sitni ljudi su lakši. Svakako bi mogli ovaj problem i promatrati kao klasifikacijski problem, ali onda se javlja nekoliko nepovoljnih situacija koje je teže riješiti. Među najpoznatijim algoritmima za regresiju su: linearna regresija, polinomna regresija, neuralne mreže, regresivna stabla, slučajne šume i dr. Kao i kod klasifikacije, neki od navedenih algoritama bit će detaljno obrađeni u daljnjem radu kao i njihovo uvođenje u nastavu.

1.3 Nenadzirano učenje

Mate i Matilda se poznaju od davnih dana vrtića. Sada idu u isti razred i upravo su učili o geometrijskim oblicima i tijelima. Kako su počeli zajedno učiti za nadolazeći ispit sjetili su se jedne svađe koje su imali u vrtiću. Bila je to žestoka svađa oko raznih geometrijskih tijela. Točnije, oko raznobojnih igračaka koje su imale oblik geometrijskih tijela. Oni su cijelu kutiju tih igračaka rasuli po podu te se odlučili pravilno posložiti igračke. Mate je počeo slagati igračke po oblicima dok je Matilda počela slagati po bojama. U nekom trenutku Mate je počeo prigovarati Matildi kako ništa ne zna i da se igračke moraju slagati po obliku dok je Matilda branila svoj način slaganja po bojama. Igračke su počele letiti sve dok jedna nije pogodila Matildu u glavu. Matilda se rasplakala i nije razgovarala s Matom sljedeća tri dana. Danas se Mate i Matilda mogu zajedno smijati dok prepričavaju tu priču.

U ovoj gornjoj priči možemo vidjeti primjer nenadziranog učenja. Mati i Matildi nitko nikad nije rekao kako treba podijeliti igračke. Nikada nisu učili kako ih podijeliti i provjeravali dali je njihov rezultat točan. Ipak su Mate i Matilda imali odgovor na taj problem, premda znatno različit. Nenadzirano učenje, za razliku od nadziranog, nikad nema parametar koji određuje krajnji rezultat (učitelja). Kod nenadziranog učenja postoje neki podaci koji možda jesu, a možda i nisu povezani. Zadatak nenadziranog učenja je da pronađe pravilnosti u danim podacima bez da zna što se eksplicitno traži. Ovakav način učenja nije lako promatrati u svakodnevnom životu, jer se pretežno događa intuitivno. Ljudi imaju taj urođeni poriv da traže pravilnosti u drugim ljudima, stvarima, događajima, oblicima itd. Nenadzirano učenje pokušava oponašati taj poriv ljudi kako bi odredilo postojeće veze među parametrima. Jedan on najjednostavnijih primjera primjene takvog načina učenja je

u online prodaji: *„Osobe koje su kupile x također su kupile y.“*. U tom slučaju prodavačke stranice pamte kupljene artikle svojih kupaca i nastoje pronaći vezu među kupljenim artiklima kako bi drugim potencijalnim kupcima dali što bolje sugestije za dodatnu kupnju.

Nenadzirano učenje je primjenjivo na dvije osnovne vrste problema: grupiranje (clustering) i asocijaciju.

Grupiranje

Primjer grupacijskog problema se opisao u priči Mate i Matilde. U toj priču imamo skup podataka, geometrijska tijela različitih boja, i zadatak je bio grupirati te podatke. Dok je taj primjer bio dosta jednostavan i rješenja očita, drugačije bi bilo kada bi se radilo o stotinama tisuća podataka koji imaju samo numeričke vrijednosti. Dok bi svakako jedna osoba mogla naći poveznice između podataka i grupirati podatke u tom moru informacija to bi ipak bio mukotrpan i spor posao. Jedan od algoritama za rješavanje problema grupiranja je *k-means* algoritam. Njega budem obradio u daljem tekstu. Postoje i razni drugi algoritmi za rješavanje ovakvog tipa problema kao što su: mean-shift clustering, DBSCAN i EM/GMM. Ti algoritmi su nešto složeniji i neće se obrađivali u ovom radu.

Asocijacija

Problem asocijacije se sastoji u tome da se u skupu podataka pronađe nekakva zavisnost između pojedinih podataka. Takav primjer asocijacije je spomenuti problem u prodaji gdje se nastoji saznati potrošačke navike kupaca (*Osobe koje su kupile x također su kupile y*). Glavna razlika ovog problema i problema grupacije je u tome što kod problema grupacije nastojimo saznati okupljaju li se podaci oko jednog mjesta dok kod asocijacije gledamo jesu li na neki način povezani.

1.4 Podržano učenje

Subota je, slobodan dan od škole i Mate se veseli. Može ići s prijateljima igrati nogomet. Okupi se Matino društvo i krene se igrati. Svako malo Mate dolazi pred vratara i puca, ali svaki put kako puca vratar obrani. Sa svakim obranjenim golom Mate postaje sve uzrujaniji. Krene novi napad Matine momčadi. Mate, još uzrujan od prošlih neuspjelih pokušaja, krene punom brzinom na protivnička vrata. Dok trči na vrata shvati da je vratar spreman na njegov napad. Mate uoči svoju uzrujanost i da će opet vratar obraniti ako pokuša pucati pa se brzo smiri. U zadnji trenutak, Mate odlučuje loptu dodati Matku koji prilazi protivničkim vratima s druge strane umjesto da puca direktno na vrata. Matko prima loptu i puca na protivnička vrata i pogađa, jer vratar nije očekivao da će Mate dodati loptu nekom drugom. Publika je oduševljena potezom i žestoko navija. Igra završi

i Matina momčad pobjeđuje. Mate se od toga dana odlučuje da će češće dodavati svojoj momčadi umjesto da puca, jer je veća šansa pobjede.

Ovaj način učenja je zasnovan na principu pokušaja i pogreške. Računalo se postavi u neku okolinu gdje može donositi odluke te više puta prolazi kroz tu okolinu. Svaki put kada napravi pogrešku nastoji učiti iz nje. Analogno, ako napravi nešo dobro onda dobiva nagradu i nastavlja ponavljati takvo ponašanje. Pogledamo li gornju priču uočavamo da Mate uzima ulogu računala, a nogomet (i njegova pravila) preuzimaju ulogu okoline. Mate prvo pokušava istrčati naprijed i ispucati loptu na gol u nadi da pogodi. Kad vidi da nekoliko puta to ne daje željeni ishod on mijenja taktiku i dodaje loptu. Matko zabija gol i publika pozitivno reagira na to. Na kraju utakmica završava pobjedom. Mate dobiva nagradu u obliku reakcije gledatelja i pobjede te se odlučuje da je dodavanje lopte bolje nego raspucavanje.

Poglavlje 2

Algoritmi strojnog učenja

U ovom poglavlju će se obraditi tri popularna algoritma strojnog učenja: stablo odlučivanja, linearna regresija i naivni Bayes. Svaki od algoritama zahtjeva određeno predznanje kako bi se razumio u cijelosti. Nastojat ću, prije obrade tih algoritama, navesti potrebno predznanje učenika. Svaki pojam bitan za razunjevanje algoritama ću obraditi pojedinačno, u odgovarajućem redoslijedu. Pojmove, definicije kao i formule nastojat ću popratiti njihovim metodičkim uvođenjem, primjerima i učeničkim postignućima.

2.1 Stablo odlučivanja

Za savladavanje ovog algoritma strojnog učenja učenicima će biti potreno predznanje iz sljedećih područja:

1. Matematika:
    - logaritamska funkcija,
    - funkcija vjerojatnosti,
    - funkcija apsolutne vrijednosti.
2. Informatika:
    - rad s listama,
    - rad sa skupovima,
    - petlja for,
    - funkcije,
    - logički izrazi,

- algoritam za traženje maksimuma.

Stablo odlučivanja je model koji ljudi pretežno koriste nesvjesno u svakodnevnom životu. Nastoje odrediti ispravan odgovor na neko pitanje. Jedan od najjednostavnijih takvih dilema, s kojem se svi mogu poistovjetiti, je procjena nositi kišobran ili ne. Većina nas će pogledati kroz prozor te pokušat odrediti koliko je oblačno i miriše li na kišu. Neki stariji od nas još se oslanjaju na kosti dok drugi uzimaju u obzir godišnje doba i koliko je sati. Sve te odluke mogu se svesti u graf kojeg zovemo *Stablo odlučivanja*. Ime je dosta intuitivno, jer taj graf poprima oblik stabla, a služi nam za donošenje odluke. Primjer takvog stabla vidi se na slici 2.1.

Slika 2.1: Jedan primjer stabla odlučivanja.

Gornji primjer je intuitivno jasan, ali pitanje kako donosimo tu odluku i zašto smo baš tako napravili stablo odlučivanja nije. Ljudi su kroz godine iskustva i nekoliko pokisnutih odjevnih predmeta, naučili što mogu očekivati. Upravo taj proces učenja je onaj koji je za nas interesantan. Razmotrimo sljedeći primjer. Zamislimo si da vodimo teren za golf i da želimo odrediti jednostavan način da našim članovima damo do znanja isplatili im se doći igrati ili ne. Koristeći zapise igara koje su se odvile i meteorološke podatke, kao što su prognoza, temperatura, vlaga i vjetar, dobiva se tablica 2.1.

Sada, kada imamo tablicu podataka, možemo pokušati iz nje izvući neke informacije. To bi bio dobar trenutak da učenicima prepustimo inicijativu da naprave svoje stablo odlučivanja koje onda možemo usporediti s računalno generiranim stablom odlučivanja.

Dan	Prognoza	Temperatura	Vlaga	Vjetar	Igrati golf
D1	Sunčano	Vruće	Visoka	Slab	Ne
D2	Sunčano	Vruće	Visoka	Jak	Ne
D3	Oblačno	Vruće	Visoka	Slab	Da
D4	Kiša	Blago	Visoka	Slab	Da
D5	Kiša	Svježe	Normalna	Slab	Da
D6	Kiša	Svježe	Normalna	Jak	Ne
D7	Oblačno	Svježe	Normalna	Jak	Da
D8	Sunčano	Blago	Visoka	Slab	Ne
D9	Sunčano	Svježe	Normalna	Slab	Da
D10	Kiša	Blago	Normalna	Slab	Da
D11	Sunčano	Blago	Normalna	Jak	Da
D12	Oblačno	Blago	Visoka	Jak	Da
D13	Oblačno	Vruće	Normalna	Slab	Da
D14	Kiša	Blago	Visoka	Jak	Ne

Tablica 2.1: Tablica podataka za igranje golfa

Aktivnost: Raste stablo

Cilj

Analizom podataka iz tablice, kroz diskusiju sa svojom grupom, stvoriti stalo odlučivanja koje što bolje opisuje dani problem te argumentirati dobiveni rezultat.

Očekivana učenička postignuća:

- Analizirati podatke u tablici prigodne za stvaranje stabla odlučivanja.
- Skicirati stablo odlučivanja sa svim označenim granama.
- Argumentirati ispravnost davanja prednosti podjele jednom atributu pored drugih.
- Surađivati s članovima grupe u pronalasku rješenja za dani problem.

Tijek aktivnosti

Nakon uvodnog primjera stabla odlučivanja te njegovog opisa i opisa njegovih sastavnih dijelova učenike se može postaviti pred izazov da stvore vlastito stablo odlučivanja. Učenike se podijeli tako da ima $4-5$ grupa. Stavi ih se pred zadatak da, koristeći tablicu

2.1, odrede što precizniji model stabla odlučivanja. Taj model se kasnije može usporediti s modelom koji se dobije primjenom algoritma za stvaranje stabla odlučivanja.

Kada svaka grupa ima nekakvo rješenje prijeđe se na demonstriranje dobivenih rješenja. Svaka grupa predstavlja svoje stablo odlučivanja te argumentira zašto su baš izabrali tu podjelu argumenata za grane stabla.

Neke od strategija rješavanja problema mogle bi biti da učenici intuitivno idu računati vjerojatnosti ili da naprave sve moguće kombinacije stabala i provjere koje od njih ima veću točnost podataka. Ovaj zadnji pristup bi mogao biti dugačak i mukotrpan te vjerojatno se neće stići provesti. U tom slučaju se može i diskutirati primjenjivost takvog algoritma.

Entropija

Nakon provedene aktivnosti: *Raste stablo* i diskusije koja je uslijedila pri njenom izvođenju može se govoriti o pojmu entropije. Kroz prošlu aktivnost učenici su sigurno uspjeli doći do nekih rješenja. Kao što sam spomenuo, neka od mogućih rješenja su pokušali odrediti stablo odlučivanja preko vjerojatnosti, odrediti sve ili većinu kombinacija te provjerili koje stablo daje najbolju točnost ili neka kombinacija tih dvaju pristupa. Učenicima bi trebao biti poznat pojam vjerojatnosti kao broj povoljnih ishoda podijeljen s brojem svih mogućih ishoda. Ako učenicima takav pojam vjerojatnosti nije poznat onda se uloži nekoliko minuta da se objasni na nekoliko primjera.

Prije, nego što se da definicija entropije, učenici moraju intuitivno shvatiti što je to uopće entropija. U tu svrhu predložio bih provođenje sljedeću aktivnost kao motivacijski primjer.

Motivacijska aktivnost: Nasumičnost

Tijek aktivnosti

Učenicima pokažemo kutiju i raznobojne predmete, recimo crvene i plave. Kažemo učenicima da će morati odrediti razinu nasumičnosti izvlačenja predmeta iz kutije. Kako bi učenici razumjeli što to znači, određivanje nasumičnost izvlačenja, na ploču možemo nacrtati brojevni pravac i dužinu koja ide od nula do jedan. Potom, u kutiju možemo staviti tri predmeta, recimo, crvene boje te postaviti pitanja:

Kolika je vjerojatnost da sada iz kutije izvučem crveni predmet i zašto? Kolika bi onda bila nasumičnost rezultata izvlačenja na skali od nula do jedan?

Očekivani odgovor bi bio da, kako ima, samo crvenih predmeta u kutiji, onda ni nema nasumičnosti. Kako nema nasumičnosti onda to znači da je jednaka nuli. U tom trenutku možemo ili staviti tri plave kuglice i pitati kolika je nasumičnost sada ili pitati učenike kako bi se postigla maksimalna nasumičnost. Nakon određivanja minimuma i maksimuma možemo početi dodavati predmete i tražiti od učenika da procjene je li nasumičnost postala

veća, manja ili je ostala ista. Nakon provedene aktivnosti možemo toj veličini nasumičnosti pridodati ime entropija te zapisati definiciju.

Definicija 2.1.1. *[17] Entropija, označena s $H(S)$ za konačan skup S, je mjera nasumičnosti u nekom eksperimentu. Ona je dana formulom:*

$$H(S) = \sum_{x \in \omega} P(x) log_2 \frac{1}{P(x)}. \tag{2.1}$$

Gde je S multiskup atributa kojeg klasificiramo a ω elementarni prostor događaja multiskupa S.

U ovom trenutku bi bilo dobro dodati još jedan primjer. Recimo da bacamo simetričnu igraću kocku sa šest strana. Imamo situaciju:

$$\omega = \{1, 2, 3, 4, 5, 6\}$$

$$P(x) = \frac{1}{6}, x \in \omega.$$

Kako je vjerojatnost dobitka svakog broja potpuno jednaka onda je mjera nasumičnosti nekog eksperimenta najveća moguća, odnosno $H(S) = \log_2 6$. Ako bi pak promatrali bacanje nesimetrične igraće kocke s šest strana kod koje vjerojatnost dobitka broja šest dana s $P(6) = \frac{1}{2}$. Entropija u tom slučaju bi bila manja od maksimalne vrijednosti, jer je nasumičnost narušena u korist pojavljivanja broja šest. Konačno, kada bi razmatrali igraću kocku koja je namještena da stalno daje broj šest onda bi entropija bila jednaka nuli, jer znamo koji ćemo broj dobiti i nema nikakve nasumičnosti.

Razmotrimo slučaj igranja golfa. Osnovna informacija koju mi moramo izvući iz toga skupa podataka je dali igrati ili ne igrati. Dakle, imamo 14 slučajnih pokusa i svaki od njih završava jednim od elementarnim događajem. Može se učenicima prepustiti ovaj izračun. U svakom slučaju zaključujemo da je:

$$S = \{Ne, Ne, Da, Da, Da, Ne, Da, Ne, Da, Da, Da, Da, Da, Ne\}$$

$$\omega = \{da, ne\}.$$

Iz danih 14 podataka imamo da su:

$$P(Da) = \frac{9}{14}$$

$$P(Ne) = \frac{5}{14}.$$

Konačno, primjenom (2.1) možemo i izračunati $H(S)$ i on iznosi:

$$H(S) = \sum_{x \in \omega} P(x) log_2 \frac{1}{P(x)}$$

$$H(S) = P(Da) log_2 \frac{1}{P(Da)} + P(Ne) log_2 \frac{1}{P(Ne)}$$

$$H(S) = \frac{9}{14} log_2 \frac{14}{9} + \frac{5}{14} log_2 \frac{14}{5}$$

$$H(S) \approx 0.94$$

Nakon što se prošlo kroz pojam, definiciju i zadatke vezane uz entropiju može se i krenuti na programerski dio. Učenicima se zada jednostavan zadatak i prepusti ih se radu.

Zadatak. Napiši funkciju **H**, i sve potrebne pomoćne funkcije, koja će kao argument primati tablicu podataka (lista koja sadrži lite) S i vratiti ukupnu entropiju podataka u toj listi. Pretpostavite da se uvijek mjeri entropija podatka koji se nalazi na zadnjem mjestu svake liste u početnoj listi S.

Primjer: Za unos tablice 2.1 u obliku liste program treba vratiti vrijednost 0.94.

Jedno moguće rješenje:

```
import math

def stupac(S, a):
    L = []
    for i in S:
        L.append(i[a])

    return L

def H(S):
    sum = 0
    P = []
    O = set()
    L = stupac(S, len(S[0])-1)

    for i in L:
        if i not in O:
            O.add(i)
```

```
    for i in O:
        P.append(L.count(i)/len(L))

    for i in P:
        sum += i * math.log(1/i, 2)
    return sum
```

Točnost dobivenih programa se može provjeriti unaprijed pripremljenim primjerima. Ovu funkciju ćemo kasnije pozivati u drugim funkcijama i programima pa bi ju bilo dobro pospremiti.

Porast informacije

Gore smo obradili pojam entropije kao mjeru nasumičnosti podataka. Kako bi smo riješili problem klasifikacije, odnosno odredili koji podatak spada u koju klasu, potrebno je smanjiti entropiju. Na istom primjeru obojenih predmeta s učenicima možemo provesti i sljedeću diskusiju. Stavimo osam crvenih i dva plava predmeta u kutiju te pitamo učenike dali je entropija velika ili mala i kolika je vjerojatnost da se izvuče crveni predmet. Počinjemo lagano dodavati više plavih predmeta u kutiju i s time povećajemo entropiju. Kada je bilo osam crvenih i dva plava predmeta u kutiji entropija je bila mala, a boja koja je prevladavala je bila crvena. Postavimo pitanja učenicima:

Što se događa kada dodajemo plave predmete u kutiju? Kada će entropija biti najveća? Što se dogodi kada dodajemo plavih predmeta iznad te granice? Koja boja onda u tom trenutku prevladava?

Cilj tih pitanja je da učenike navodimo do zaključka da što je entropija manja to znači da pouzdanije možemo predvidit rezultat (prevladava određena boja). Što više prevladava određena boja to više možemo biti sigurni da će se izvući određena klasa predmeta, odnosno boja. Sljedeći zadatak nam onda postaje smanjivanje entropije koristeći podatke koje imamo kako bi odredili klasu. Tu u igru dolazi pojam porast informacije.

Definicija 2.1.2. *Porast informacije je razlika između entropije multiskupa i sume entropija multiskupa podijeljenog po određenom atributu i dana je s [17]formulom:*

$$IG(S, A) = H(S) - \sum_{v \in \omega_A} \frac{|S_v|}{|S|} H(S_v), \tag{2.2}$$

gdje je A atribut kojeg ne klasificiramo, ω_A elementarni prostor događaja za atribut A i S_v podmultiskup multiskupa S za kojeg atribut A ima vrijednost v, $v \in \omega_A$.

Ovisno o parametru po kojem želimo stvoriti sljedeće grane stabla, entropija će se smanjiti više ili manje. Pogledajmo na primjeru igranja golfa. Recimo da nas zanima

dobit informacija od atributa: Vjetar. Gledajući tablicu vidimo da su elementarni događaji za odabrani atribut: *Slab* i *Jak* pa je $\omega_A = \{Slab, Jak\}$. Od 14 danih primjera 8 je slab vjetar, a 6 je jak vjetar pa su $|S_{Slab}| = 8$ i $|S_{Jak}| = 6$. Kako smo od prije izračunali $H(S)$, uvrštavanjem navedenih podataka u (2.2) dobivamo tablice 2.2 i 2.3.

$$IG(S, Vjetar) = 0.94 - \frac{8}{14}H(S_{Slab}) - \frac{6}{14}H(S_{Jak}).$$

Konačno, trebamo odredit $H(S_{Slab})$ i $H(S_{Jak})$. Za to prvo moramo podijeliti početnu tablicu u dvije manje i to tako da su u jednoj tablici sve vrijednosti parametra $Vjetar = Slab$, a u drugoj $Vjetar = Jak$. Dobivamo:

Dan	Prognoza	Temperatura	Vlaga	Vjetar	Igrati golf
D1	Sunčano	Vruće	Visoka	Slab	Ne
D3	Oblačno	Vruće	Visoka	Slab	Da
D4	Kiša	Blago	Visoka	Slab	Da
D5	Kiša	Svježe	Normalna	Slab	Da
D8	Sunčano	Blago	Visoka	Slab	Ne
D9	Sunčano	Svježe	Normalna	Slab	Da
D10	Kiša	Blago	Normalna	Slab	Da
D13	Oblačno	Vruće	Normalna	Slab	Da

Tablica 2.2: Tablica podataka za igranje golfa kada je vjetar slab.

Dan	Prognoza	Temperatura	Vlaga	Vjetar	Igrati golf
D2	Sunčano	Vruće	Visoka	Jak	Ne
D6	Kiša	Svježe	Normalna	Jak	Ne
D7	Oblačno	Svježe	Normalna	Jak	Da
D11	Sunčano	Blago	Normalna	Jak	Da
D12	Oblačno	Blago	Visoka	Jak	Da
D14	Kiša	Blago	Visoka	Jak	Ne

Tablica 2.3: Tablica podataka za igranje golfa kada je vjetar jak.

Iz tablica 2.2 i 2.3 izvlačimo informaciju da je $S_{Slab} = \{Ne, Da, Da, Da, Ne, Da, Da, Da\}$ i $S_{Jak} = \{Ne, Ne, Da, Da, Da, Ne\}$. Iz tih dvaju skupova možemo izračunati $H(S_{Slab})$ i $H(S_{Jak})$ i oni iznose:

$$H(S_{Slab}) \approx 0.811$$

i

$$H(S_{Jak}) = 1.$$

Sada kada imamo i te zadnje potrebne podatke možemo i izračunati porast informacije $IG(S, Vjetar)$ i on iznosi:

$$IG(S, Vjetar) \approx 0.048.$$

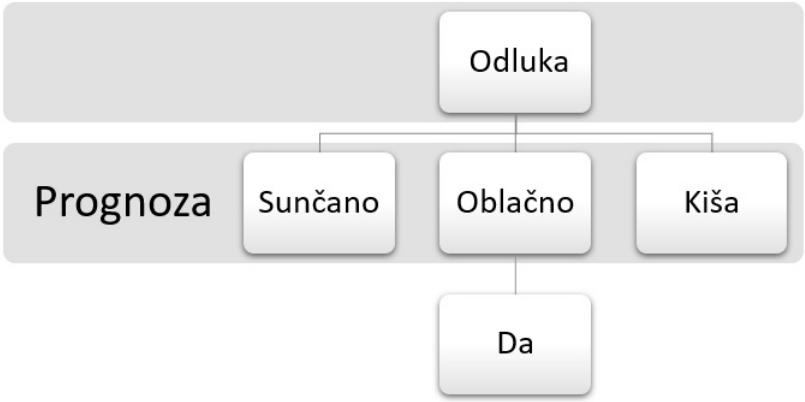
Analogno možemo ponoviti postupak za atribute: prognoza, temperatura i vlaga i dobiti sljedeće vrijednosti:

$$IG(S, Prognoza) \approx 0.246,$$
$$IG(S, Temperatura) \approx 0.029$$

i

$$IG(S, Vlaga) \approx 0.151.$$

Iz tih rezultata vidimo da najveći porast informacije dobivamo od atributa *Prognoza*. Dakle, u stablu odlučivanja ćemo prvo dijeliti po atributu *Prognoza* te s time dobiti izgled stabla kao na slici 2.2.



Slika 2.2: Stablo odlučivanja za tablicu igranja golfa nakon prve podjele.

Kod grane *Oblačno* se odmah upisalo *Da*, jer ako pogledamo sve slučajeve u kojima je prognoza oblačna uočit ćemo da, se svaki put igrao golf. Kada bi nastavili tražiti entropiju i

porast informacija na novim tablicama u kojima se, samo gledaju podaci kada je prognoza bila sunčana, oblačna ili kiša, onda bi dobili sljedeću razinu u stablu odlučivanja. I tako možemo nastaviti računati sve dok ne istrošimo sve atribute. Na kraju bi dobili potpuno stablo odlučivanja. Prije, nego što se može napraviti potpuno stablo odlučivanja, treba se napisati program koji će određivati najveći porast informacija.

Zadatak

Napišite funkciju **IG**, i sve potrebne pomoćne funkcije, koja će kao argument primati tablicu podataka (lista koja sadrži liste), a vraćati indeks onog atributa čiji je porast informacija najveći. Pretpostavite da se uvijek mjeri entropija podatka koji se nalazi na zadnjem mjestu svake liste u početnoj listi S.

Primjer: za unos tablice 2.1, bez oznake dana, funkcija IG treba vratiti vrijednost 0, jer atribut*Prognoza* daje najveću dobit informacija, a on se nalazi na mjestu s indeksom nula.

Jedno moguće rješenje je:

```
def IG(S):
    max = [0,-1]
    entropija = H(S)
    O = set()
    L1 = []
    L2 = []
    L3 = []

    for i in range (len(S[0])-1):

        L1 = stupac(S, i)
        for j in L1:
            if j not in O:
                O.add(j)
        for j in O:
            for k in S:
                if k[i] == j:
                    L2.append(k)

            entropija -= (len(L2)/len(S))*H(L2)
            L2.clear()

        L3.append(entropija)
        entropija = H(S)
        L1.clear()
        O.clear()
    for i in range(len(L3)):
        if max[1] < L3[i]:
            max = [i, L3[i]]

    return max[0]
```

Python i stablo odlučivanja

Pored određenog pojma entropije i porasta informacije te njihovog programskog rješenja može se dobiti i stablo odlučivanja. Ono se dobiva tako da se proces traženja entropije i porasta informacije ponavlja. Pitanje se samo javlja do kada se treba ponavljati taj proces. Jasno je da kod ovakvog procesa imamo pojavu rekurzije, ponavlja se iznova jedan te isti proces dok ne dođe do nekog zaustavnog uvjeta. Kroz dijalog nastavnik treba učenike dovesti do zaključka da se radi o rekurzivnom ponašanju te ih potom pitati koji bi bili zaustavni uvjeti. Jedan zaustavni uvjet se vidio u gornjem primjeru kada je atribut *Prognoza* jednak *Oblačno*. Razlog tome je, kao što je bilo rečeno, taj što u tablici svaki put kada je prognoza oblačna igra se golf. Ako se svaki put igra golf onda je jasno da je razina nasumičnosti, odnosno entropija, jednaka nuli. Drugi zaustavni uvjet je kada je ostao samo jedan atribut po kojem možemo klasificirati. U trenutku kada se ima samo jedan atribut, po kojem se klasificira, nema smisla gledati porast informacije, jer taj atribut sigurno ima najveći porast. Tada treba pogledati za koje vrijednosti se dobiva koja klasa.

Zadatak. Koristeći funkcije za entropiju i porast informacija napišite funkciju **stablo**, i sve potrebne pomoćne funkcije, koja će kao argument primati tablicu podataka (lista koja sadrži liste), a zapis stabla odlučivanja u nekom obliku.

Jedno od mogućih rješenja je:

```
def IG(S):
def stablo(S, T):
    L1 = []
    O = set()
    X = IG(S)

    if len(S) == 2 and H(S) != 0:
        L1 = stupac(S, 1)

        for i in L:
            if i not in O:
                O.add(i)

        X = 0
        s = O.pop()
        O.add(s)

        for i in O:
            if X < S.count(i):
```

```
                s = i

        T.append(s)
        return

    elif H(S) == 0:
        T.append(S[0][len(S[0])-1])
        return

    L1 = stupac(S, X)

    for i in L1:
        if i not in O:
            O.add(i)

    for i in O:
        T.append(i)

    for i in O:
        stablo(nova(S,i,X),T)

    return T
```

Kroz pisanje funkcije učenici će sigurno doći do jednog problema. Kada se dolazi do zadnje razine stabla, može se dogoditi da su sve klase jednako moguće. Za taj slučaj nije strogo definirano što se dogodi. Jedno od rješenja je da se uzme jedna klasa nasumice ili se može dodati dodatna oznaka koja obilježava da se radi o slučaju gdje je svaka klasa jednako moguća.

Provjera znanja

Neki od mogućih načina provjere znanja ovog gradiva uključuju:

- Opis pojmova: stablo odlučivanja, entropija i porast informacija vlastitim riječima uz primjere.
- Konstrukcija stabla odlučivanja na jednostavnim primjerima.
- Procjena razine entropije za dane podatke i njezin izračun.
- Ručni izračun porasta informacije za dane podatke.

- Navođenje primjera problema gdje bi se algoritam mogao koristiti.

2.2 Linearna regresija

Za savladavanje ovog algoritma strojnog učenja učenicima će biti potreno predznanje iz sljedećih područja:

1. Matematika:
    - rad sa matricama,
    - derivacije,
    - određivanje ekstrema funkcija preko derivacija,
    - parcijalne derivacije.
2. Informatika:
    - rad s listama,
    - petlja for,
    - funkcije,
    - logički izrazi.

Za razliku od stabla odlučivanja, [6]linearna regresija služi za predviđanje jedne ili više realnih varijabli, $t \in \mathbb{R}^n, n \in \mathbb{N}$, ako su poznate vrijednosti k-dimenzionalnog vektora X ulaznih vrijednosti. U ovom radu obradit će se samo linearna regresija koja predviđa vrijednost jedne realne varijable. Odnosno, razmatrat će se linearne regresije oblika

$$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}.$$

Prisjetimo se primjera procjene težine kojeg smo naveli. U tome primjeru imamo dvije varijable, visinu i stas, iz kojih moramo predvidjeti težinu. Odnosno, imamo dvodimenzionalan vektor

$$X = (visina, stas), visina \in \mathbb{R}, stas \in \{1, 2, 3\},$$

gdje brojevi $1, 2$ i 3 predstavljaju sitan, normalan i krupan stas, i traženu varijablu težine $t \in \mathbb{R}$. Ovaj primjer je dosta prikladan, jer se nečim mjerljivim i očitim lako pokaže da su ljudi skloni koristiti linearnu regresiju na intuitivnoj razini kako bi predvidili željene ishode. Primjer toga možemo vidjeti kod studenata i učenika koji imaju pristup ispitima prošlih godina. Njihova osnovna strategija učenja je da pogledaju ispite prošlih godina, odrede tipove zadataka koji su se pojavljivali u njima, rastave te zadatke u podskupine

gradiva i po tome na što je stavljen naglasak na predavanju odrede što je najvjerojatnije da se pojavi u nadolazećem ispitu. Preko te analize učenici si mogu lakše odrediti strategiju učenja, odnosno prioritete učenja.

Pogađanje težine

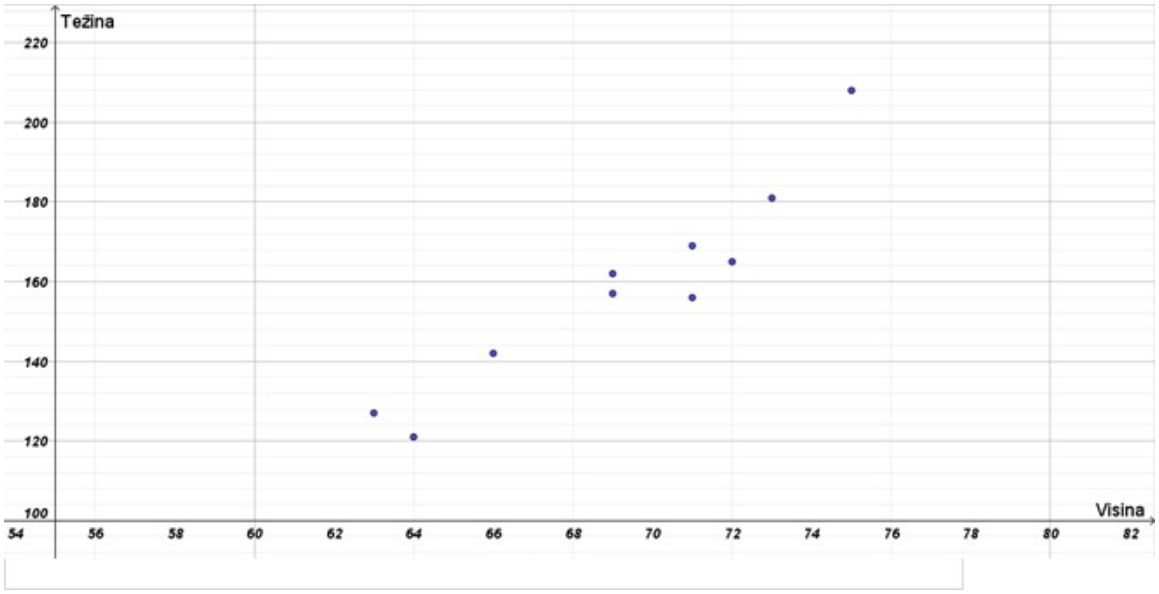
Dobara motivacija, prije samog uvoda ovog gradiva, bi bila da se učenike poreda u red i izabere se jednog učenika kojem se zada zadatak da procijeni težinu svake osobe u redu. Uz možda malo neugodnosti na početku učenik bi počeo procjenjivati težine svojih prijatelja. Nakon nekoliko procjena učenika se može pitati kako je došao do tih zaključaka. Učenik će vjerojatno reći da je naprosto pogledao i procijenio. U tom bi se trenutku učenika trebalo pitati na što je točno obazirao pažnju. Kroz razgovor bi se u nekom trenutku došlo do zaključka da je učenik svoju procjenu bazirao po obliku i eventualno visini osobe. U tome trenutku se učenicima kaže da je on zapravo intuitivno koristio ono o čemu će se pričati, odnosno, linearnu regresiju.

Razmotrimo pojednostavljeni primjer procjene težine osobe. Raspolažemo s podatcima koji nam govore o visini pojedinca i njegovoj težini. Sljedeća 2.4tablica podataka izražava visinu u inčima i težinu u funtama (Ove podatke nastavnik može prilagoditi nastavi uz pomoć nastavnika tjelesnog i zdravstvenog odgoja koristeći podatke dobivene mjerenjem učenika):

Vis	**Tež**
63	127
64	121
66	142
69	157
69	162
71	156
71	169
72	165
73	181
75	208

Tablica 2.4: Tablica visina i težina učenika.

Dane podatke se može prikazati grafom. Taj dio posla nastavnik može prepustiti učenicima da naprave koristeći program koji njima odgovara. Moja preporuka bi bila neki program dinamičke geometrije poput GeoGebra. Nakon nekog vremena svi bi trebali imati nekakav graf kao na slici 2.3.



Slika 2.3: Grafički prikaz podataka iz tablice 2.4.

Iz te slike vidi se da ima nekakva veza između visine i težine. Čini se da, što je veća visina to je i težina veća, što je bilo očekivano. U najboljem slučaju sve točke, dobivene u grafu, će biti kolinearne i možemo uspostaviti potpuno linearnu ovisnost svih podataka, ali to nije tako. Učenike postavimo pred problem da odrede pravac koji će najbolje procijeniti sve dane podatke. Velika je vjerojatnost da je svaki učenik nacrtao drugačiji pravac. Nekoliko tih pravaca se može nacrtati, kao na slici 2.4, i učenicima postaviti pitanje kako odrediti koji pravac najbolje procjenjuje podatke.

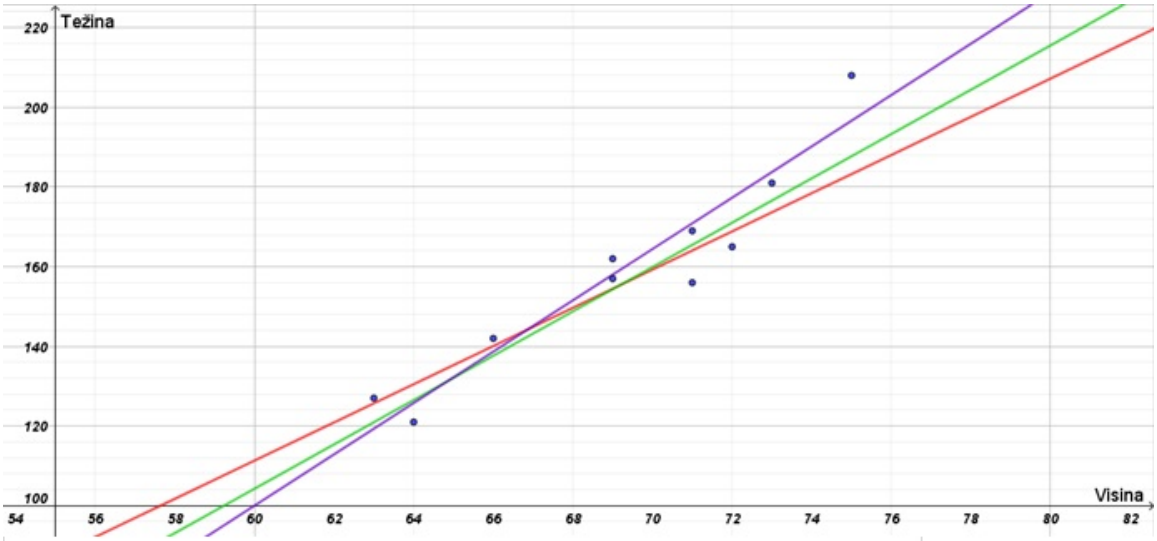
Nacrtani pravci su pravci regresije. Neki bolje, a neki lošije procjenjuju dane podatke. Nama je cilj, naravno, naći pravac regresije koji najbolje procjenjuje dane podatke. Znamo da se radi o pravcu, a poznata nam je i jednadžba pravca koja glasi:

$$y = a + bx. \tag{2.3}$$

Za naše potrebe jednadžbu (2.3) zvat ćemo hipoteza. Hipoteza je linearna funkcija koja procjenjuje dane podatke. Osim toga trebamo i funkciju pogreške hipoteze koju zadajemo s:

$$J(a,b) = \frac{1}{2n}\sum_{i=1}^{n}(a + bx^{(i)} - y^{(i)})^2. \tag{2.4}$$

Gdje je $n \in \mathbb{N}$ broj podataka, $x^{(i)}$ je i-ti podatak po redu u podacima, a $y^{(i)}$ i-ta vrijednos po redu. Za primjer iz tablice 2.4 vrijedilo bi: $x^{(1)} = 63$ i $y^{(1)} = 127$, $x^{(5)} = 69$ i $y^{(5)} = 162$



Slika 2.4: Nekoliko pravaca koji procjenjuju ovisnost visine i težine.

i $x^{(7)} = 71$ i $y^{(7)} = 169$ itd. Nama je u cilju postići što manju pogrešku. Kako bi se našla najmanja pogreška treba se odrediti minimum funkcije pogreške. Odnosno trebaju se odrediti takvi a i b da $J(a, b)$ bude najmanja. U tu svrhu će se koristiti metoda gradijentnog spusta.

Za razliku od direktnog traženja minimuma funkcije, gradijentni spust, se postepeno približava minimumu funkcije. Početne vrijednosti a i b se postave na neke proizvoljne brojeve te se iterativnom metodom dolazi do njihovih pravih vrijednosti koristeći sljedeće formule:

$$a := a - \alpha \frac{\partial}{\partial a} J(a, b),$$

$$b := b - \alpha \frac{\partial}{\partial b} J(a, b).$$

Odnosno kada se parcijalno deriviraju dobiju se sljedeće jednakosti:

$$a := a - \alpha \frac{1}{n} \sum_{i=1}^{n} (a + bx^{(i)} - y^{(i)}), \tag{2.5}$$

$$b := b - \alpha \frac{1}{n} \sum_{i=1}^{n} (a + bx^{(i)} - y^{(i)}) x^{(i)}. \tag{2.6}$$

Koeficijent α predstavlja brzinu gradijentnog spusta. Kao i početne vrijednosti koeficijenata pravca regresije i α se bira proizvoljno. Ipak pri izboru brzine gradijentnog spusta treba se uzeti u obzir nekoliko stvari. Ako je brzina gradijentnog spusta premala onda bi izračun mogao trajati dugo. Na drugoj strani spektra, ako je brzina gradijentnog spusta prevelika može se dogoditi da postupak preskoči minimum i nikada ga ne dosegne.

Izračun koeficijenata može biti mukotrpan posao pogotovo ako se ne izabere dobar α ili ako je broj potrebnih iteracija velik. Zato je najbolje prepustiti računalu da izračuna koeficijente. Naravno, učenicima prepuštamo da napišu potrebni program i provedu ispitivanje na danim podacima.

Zadatak.

Napiši funkciju **pravac_regresije**, i sve potrebne pomoćne funkcije, koja će primati argumente S, α i *it* gdje je S tablica podataka (lista koja sadrži liste), α je brzina gradijentnog spusta, a *it* broj iteracija. Na početku koeficijente a i b postavite na jedan.

Jedno od mogućih rješenja je:

```
def pravac_regresije (S, alpha, it):

    a = 1
    b = 1

    for i in range (it):

        s_1 = 0
        s_2 = 0

        for j in S:
            s_1 = s_1 + a + b * j[0] - j[1]

        a_1 = a - alpha * (1/len(S)) * s_1

        for j in S:
            s_2 = s_2 + (a + b * j[0] - j[1]) * j[0]

        b_1 = b - alpha * (1/len(S)) * s_2

        a = a_1
        b = b_1

    return a, b
```

Nakon napisanog programa treba se provjeriti koliko je točan i radi li. U tu svrhu dobro je pokrenuti program na podacima iz primjera. Za te dane podatke [21]znamo da su $a \approx -267$ i $b \approx 6.14$. U gornjoj varijanti programa, kako bi se dobili približni rezultati bit će potrebno postaviti $\alpha = 0.00041$, a $it = 5000000$ kako bi se dobilo približno dobre rezultate od $a = -265.78$ i $b = 6.12$. Ako se α postavi veća onda se ubrzo dobije rezultat NaN što znači da je program dosegao svoju granicu. Na ovom primjeru se može lako vidjeti koliko sporo koeficijenti konvergiraju prema konačnom rezultatu. Bilo bi dobro s učenicima proći kroz nekoliko različitih unosa da se vidi kako izbor α i it utječe na izvršavanje programa i njegovu točnost. Nakon što učenici dobiju neko svoje približno rješenje mogu ga odmah i usporediti sa službenim rješenjem u programu dinamičke geometrije kao što je GeoGebra. Tamo se može lako vidjeti da se dobiveni rezultat skoro pa poklapa sa službenim rješenjem.

Generalizacija gradijentnog spusta

Kroz primjer visine i težine učenika se riješio problem predviđanja jedne varijable s jednom ulaznom vrijednošću. Nakon tog riješenog problema treba se generalizirati postupak i prijeći se na problem jedne varijable i $n \in \mathbb{N}$ ulaznih vrijednosti. Kako bi taj dio jednostavnije obradili treba se ponoviti gradivo linearne algebre i matrica. Razmotrimo sljedeći problem procjene vrijednosti stambenih prostorija, tablica 2.5, površina je dana u kvadratnim stopama, starost objekta u godinama i cijena u tisućama dolara.

Površina	**Broj spavaćih soba**	**Broj katova**	**Starost objekta**	**Cijena**
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Tablica 2.5: Tablica vrijednosti stanbenih prostorija.

Kao što vidimo, imamo četiri ulazne vrijednosti: površinu, broj spavaćih soba, broj katova i starost objekta preko kojih trebamo odrediti jednu varijablu, cijenu kuće. Sada, umjesto da tražimo regresijski pravac, mi tražimo općenitu linearnu regresiju. Kod te općenite linearne regresije se traži hiperravnina koja najbolje aproksimira dane podatke. Kao i kod pravca regresije, trebamo odrediti najbolju hipotezu. Ta hipoteza, za općeniti slučaj, dana je [20]sa:

$$h_\theta(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n. \tag{2.7}$$

Gdje je $X = (x_0, x_1, x_2, \ldots, x_n)$ vektor ulaznih vrijednosti, $\Theta = (\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$ vektor pripadnih koeficijenata ulaznih vrijednosti i n broj ulaznih vrijednosti. Nadalje, oznakama x^i i x^i_j označavamo ulazne vrijednosti i-tog primjera i vrijednost ulazne vrijednosti j u i-tom primjeru. S učenicima bi svakako bilo dobro proći ove pojmove na danom primjeru. Tako da jednadžba (2.7) za dani primjer glasi:

$$h_\theta(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4.$$

Tu je dobro napomenuti da θ_0 ustvari predstavlja slobodni koeficijent i da se uzima $x_0 = 1$ kako se ne bi poremetio slobodni član. Nadalje, treba proći kroz neke primjere određivanja kao: Koje vrijednosti odgovaraju: $X^{(2)}, X^{(3)}, x^3_{(2)}, x^2_{(1)}, x^1_{(4(}$. Za dane primjere odgovori bi bili: $X^{(2)} = (1416, 3, 2, 40), X^{(3)} = (1534, 3, 2, 30), x^3_{(2)} = 2, x^2_{(1)} = 5, x^1_{(4)} = 852$. Učenicu bi trebali savladati snalaženje preko tih indeksa prije, nego što se nastavi dalje.

Kao i kod slučaja jedne varijable i jedne ulazne vrijednosti i tu trebamo odrediti funkciju pogreške hipoteze. Ona je dana s:

$$J(\Theta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(X^{(i)}) - Y^{(i)})^2. \tag{2.8}$$

Gdje je $Y^{(i)}$ vrjednost i-te varijable za i-te ulazne vrijednosti $X^{(i)}$. Postupak traženja minimuma funkcije $J(\Theta)$ je jednak kao i prije. Trebaju se odrediti pojedini koeficijenti vektora Θ takvi da je $J(\Theta)$ minimalna. To se naravno postiže postupkom gradijentnog spusta, samo što se ovaj put postupak obavlja za svaki θ_i. Nova vrijednost svakog θ_i se dobiva deriviranjem i dane su formulama:

$$\theta_0 := \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(X^{(i)}) - Y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(X^{(i)}) - Y^{(i)}) x^{(i)}_1$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(X^{(i)}) - Y^{(i)}) x^{(i)}_2$$

$$\vdots$$

$$\theta_n := \theta_n - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_\theta(X^{(i)}) - Y^{(i)}) x^{(i)}_n$$

Kao i kod početnog primjera α je brzina gradijentnog spusta koja se treba ručno odrediti.

Zadatak.

Napiši funkciju *linearna_regresija*, i sve potrebne pomoćne funkcije, koja će primati argumente S, α i *it* gdje je S tablica podataka (lista koja sadrži liste), α je brzina gradijentnog spusta, a *it* broj iteracija. Početne koeficijente postavite na nulu.

Jedno od mogućih rješenja je:

```
def zbroj(S, koef, x):

    zbr = 0
    a = 0

    if x == 0:

        for i in S:
            for j in range (len(koef)):
                if j == 0:
                    a += koef[j]
                else:
                    a += koef[j]*i[j-1]

            a -= i[len(S[0])-1]
            zbr += a
            a = 0

        return zbr

    else:

        for i in S:
            for j in range (len(koef)):
                if j == 0:
                    a += koef[j]
                else:
                    a += koef[j]*i[j-1]

            a -= i[len(S[0])-1]
            a = a*i[x-1]

            zbr += a
            a = 0
```

```
        return zbr

def linearna_regresija(S, alpha, it):

    new_coef = [1]*len(S[0])
    old_coef = [1]*len(S[0])
    acc = 1

    for j in range (it):

        for i in range (len(S[0])):
            new_coef[i] = old_coef[i] - alpha*(1/len(S))*
               zbroj(S, old_coef, i)

        old_coef = new_coef.copy()

    return new_coef
```

Svoja rješenja učenici mogu provjeriti na nekim od primjera sa ove [1]poveznice na internetu. Jedan od tamo danih primjera je tablica 2.6, koja sadrži dvije ulazne vrijednosti i jednu varijablu. Tamo su odmah i dana rješenja, $\theta_0 = 8.03, \theta 1 = -3.57$ i $\theta_2 = 0.96$, koja mogu poslužiti za provjeru. Uz malo namještanja početnih parametara poziva funkcije učenici mogu zaključiti da je za ove podatke potrebno znatno manje iteracija i da brzina gradijentnog spusta može biti veća bez da se žrtvuje preciznost dobivenog rezultata. Čini se da napisani program bolje funkcionira sa više ulaznih vrijednosti nego sa jednom ulaznom vrijednosti i jednom varijablom. Na gore navedenoj poveznici ima još nekoliko primjera na kojima se može provjeriti ispravnost napisanog programa.

Provjera znanja

Neki od mogućih načina provjere znanja ovog gradiva uključuju:

- Opis pojmova: regresijski pravac, linearna regresija, hipoteza, funkcija pogreške hipoteze i brzina gradijentnog spusta vlastitim riječima uz primjere.
- Određivanje prikladnog broja iteracija i brzine gradijentnog spusta za rješavanje konkretnog problema.
- Navođenje primjera problema gdje bi se algoritam mogao koristiti.

y	x_1	x_2
9.29	1	4
12.67	2	12
12.41	3	16
0.38	4	8
20.77	5	32
9.52	6	24
2.38	7	20
7.46	8	28

Tablica 2.6: Tablica podataka za provjeru programa *linearna_regresija*.

2.3 Naivni Bayes

Za savladavanje ovog algoritma strojnog učenja učenicima će biti potreno predznanje iz sljedećih područja:

1. Matematika:
    - funkcija vjerojatnosti,
    - Bayesova formula,
    - pravilo umnoška vjerojatnosti,
    - nezavisnost događaja.
2. Informatika:
    - rad s listama,
    - rad sa skupovima,
    - petlja for,
    - funkcije,
    - logički izrazi,
    - traženje maksimuma.

Kao što bi ime dalo sugerirati, ovaj algoritam se uvelike oslanja na teoriju vjerojatnosti, a specifično na Bayesov teorem. Algoritam rješava klasifikacijske probleme u strojnom učenju. [18] Naivni Bayes je jedan od najpraktičnijih algoritama klasificiranja i u većini slučajeva je jednako dobar, a ponekad i bolji, od drugih algoritama kao što su stabla

odlučivanja i neuralne mreže. Kao i kod stabla odlučivanja tu se želi odrediti pripadajuća klasa sa što većom vjerojatnošću za neki dani unos.

Razmotrimo obrađeni primjer igranja golfa. Ulazne vrijednosti su prognoza, temperatura, vlaga i vjetar te dvije moguće klase *Da* i *Ne*. Treba se odrediti veća od vjerojatnosti ishoda *Da* i *Ne* uz uvjet *A*, gdje je *A* neka kombinacija prognoze, temperature, vlage i vjetra.

Gledano s perspektive vjerojatnosti, *A* je događaj, a *Da* i *Ne* su hipoteze. Dakle, trebamo odrediti vjerojatnost hipoteze *Da* uz uvjet da se dogodio *A* i vjerojatnost hipoteze *Ne* uz uvjet da se dogodio *A*. Odnosno, treba se odrediti koja je veća od vjerojatnosti:

$$P(Da|A) \tag{2.9}$$

i

$$P(Ne|A). \tag{2.10}$$

Kako bi smo odredili gore navedene vjerojatnosti moramo koristiti Bayesov teorem koji glasi:

Teorem 2.3.1. *Neka je* $H_i, i \in \{1, 2, \ldots, n\}, n \in \mathbb{N}$*, potpun sustav događaja i vjerojatnosnom prostoru* (ω, F, P) *i neka je* $A \in F$ *događaj takav da je* $P(A) > 0$*. Tada za svaki* $i \in \{1, 2, \ldots, n\}, n \in \mathbb{N}$*, vrijedi:*

$$P(H_i|A) = \frac{P(H_i)P(A|H_i)}{P(A)}. \tag{2.11}$$

Sada se mogu odrediti vjerojatnosti iz primjera sa igranjem golfa. Iz (2.9), (2.10) i (2.11)dobiva se:

$$P(Da|pro \wedge tem \wedge vla \wedge vje) = \frac{P(Da)P(pro \wedge tem \wedge vla \wedge vje|Da)}{P(pro \wedge tem \wedge vla \wedge vje)}$$

$$P(Ne|pro \wedge tem \wedge vla \wedge vje) = \frac{P(Ne)P(pro \wedge tem \wedge vla \wedge vje|Ne)}{P(pro \wedge tem \wedge vla \wedge vje)}.$$

Kod gornjih jednadžba zamjenjujemo *A* s $pro \wedge tem \wedge vla \wedge vje$, jer kao što je rečeno, *A* predstavlja, samo neku, od mogućih kombinacija tih ulaznih vrijednosti. Kako je vjerojatnost za svaku kombinaciju, $P(pro \wedge tem \wedge vla \wedge vje)$, jednako vjerojatna, a mi tražimo maksimum između dvije hipoteze, možemo taj dio izbaciti iz izračuna te odrediti veću vrijednost koristeći izraz:

$$max\{P(Da)P(pro \wedge tem \wedge vla \wedge vje|Da), P(Ne)P(pro \wedge tem \wedge vla \wedge vje|Ne)\} \tag{2.12}$$

Razmotrimo sada vjerojatnosti koje se pojavljuju u gornjem izrazu. Vjerojatnosti $P(Da)$ i $P(Ne)$ je lako provjeriti, jer samo moramo odrediti frekvenciju pojavljivanja od Da i Ne u danim podacima. Kod uvjetnih vjerojatnosti to je malo teže, jer postoji $3 \cdot 3 \cdot 2 \cdot 2 = 36$ kombinacija prognoze, temperature, vlage i vjetra. Problem je u tome da imamo svega 14 primjera u podacima što nije ni blizu da dobijemo dobru procjenu vjerojatnosti. Trebali bi smo imati znatno više podataka kako bi mogli odrediti te vjerojatnosti.

U ovom trenutku dolazi osnovna pretpostavka algoritma naivnog Bayesa, a ta je da su ulazne vrijednosti nezavisne jedna o drugoj. Znamo da, po pravilu produkta vjerojatnosti za nezavisne događaje vrijedi:

$$P(A_1 \wedge A_2 \wedge \ldots \wedge A_n) = \prod_{i=1}^{n} P(A_i). \tag{2.13}$$

Za slučaj kada tražimo vrijednost izraza $P(A_1 \wedge A_2 \wedge \ldots \wedge A_n|H)$, kao što je slučaj kod naivnog Bayes, uvrštavanjem jednakosti (2.13) dobivamo izraz:

$$P(A_1 \wedge A_2 \wedge \ldots \wedge A_n|H) = \prod_{i=1}^{n} P(A_i|H). \tag{2.14}$$

Konačno, primjenom (2.14) na (2.12) vidimo da u traženom zadatku trebamo odrediti:

$$max\{P(Da)P(pro|Da)P(tem|Da)P(vla|Da)P(vje|Da), P(Ne)PP(pro|Da)P(tem|Da)P(vla|Da)P(vje|Da)\} \tag{2.15}$$

Pogledajmo sljedeći zadatak:

Zadatak. Vlasnik golf terena želi svojim članovima dati obavijest o dostupnosti terena za idući dan. Za dane podatke o sutrašnjem danu odredite koju objavu bi vlasnik golf terena trebao staviti na svoju internet stranicu ako zna da će sutra biti sunčano svježe vrijeme s visokom vlagom i jakim vjetrom i poznati su mu podaci prošla dva tjedna po tablici 2.1.

Prvo se trebaju odrediti sljedeće vjerojatnosti:

$$P(Da), \qquad P(Ne),$$

$$P(Suncano|Da), \quad P(Svjeze|Da), \quad P(Visoka|Da), \quad P(Jak|Da),$$

$$P(Suncano|Ne), \quad P(Svjeze|Ne), \quad P(Visoka|Ne), \quad \text{i } P(Jak|Ne).$$

Sve te vjerojatnosti se mogu lako isčitat iz dane tablice.

$$P(Da) = \frac{9}{14}, \qquad P(Ne) = \frac{5}{14},$$

$$P(S\,uncano|Da) = \frac{2}{9}, \quad P(S\,vjeze|Da) = \frac{3}{9}, \quad P(Visoka|Da) = \frac{3}{9}, \quad P(Jak|Da) = \frac{3}{9},$$

$$P(S\,uncano|Ne) = \frac{3}{5}, \quad P(S\,vjeze|Ne) = \frac{1}{5}, \quad P(Visoka|Ne) = \frac{4}{5}, \quad \text{i } P(Jak|Ne) = \frac{3}{5}.$$

Uvrštavanjem gornjih rezultata u (2.15) dobivamo:

$$P(S\,uncano \wedge S\,vjeze \wedge Visoka \wedge Jak|Da) = \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} = \frac{54}{6561}$$

$$P(S\,uncano \wedge S\,vjeze \wedge Visoka \wedge Jak|Ne) = \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} = \frac{36}{625}$$

$$max\left\{\frac{9}{14} \cdot \frac{54}{6561}, \frac{5}{14} \cdot \frac{36}{625}\right\} = max\{0.0053, 0.0206\} = 0.0206$$

Kako vrijednost 0.0206 odgovara uvjetu da se ne igra golf zaključujemo da bi poruka na internet stranici golf terena trebala glasiti da golf teren neće biti dostupan idućeg dana.

Programiranje i primjena

Sam koncept ovog algoritma je dosta intuitivan. Za svaku klasu odredimo njenu vjerojatnost uz uvjet da se dogodio neki događaj te ona klasa koja ima najveću vjerojatnost je vjerojatno ispravan izbor. Najveći izazov učenicima tu predstavlja poznavanje matematičkih pojmova vjerojatnosti. Po potrebi se s učenicima može proći kroz potrebne definicije i primjere koji čine "matematički dio" ovog algoritma. Za strojno učenje je ipak više interesantno programsko rješenje kao i primjena tog algoritma.

Zadatak.

Napiši funkciju *bayes*, i sve potrebne pomoćne funkcije, koja će kao argument primati tablicu podataka (lista koja sadrži liste) S preko koje odlučuje kako će klasificirati podatak, a zadan kao lista koristeći Naivni Bayes.

Jedno moguće rješenje je:

```
def stupac(S, a):
    L = []
    for i in S:
        L.append(i[a])

    return L

def nova(S, atr):
```

```
    N = []

    for i in S:
        if i[len(S[0]) - 1] == atr:
            N.append(i)

    return N

def bayes(S, x):
    L1 = stupac(S, len(S[0]) - 1)
    L2 = []
    L3 = []
    cnt = 0
    prod = 1

    O = set ()

    for i in L1:
        if i not in O:
            O.add(i)

    for i in O:
        L2 = nova(S, i)

        for j in x:
            for k in L2:
                if j in k:
                    cnt+=1

            prod *= cnt/len(L2)
            cnt = 0

        L3.append([i,prod * (len(L2)/len(S))])
        prod = 1
        L2.clear()

    for i in range (len(L3)):
        if L3[i][1] > cnt:
            cnt = L3[i][1]
```

```
            L2=[L3[i][0]]

    return L2[0]
```

Ispravnost svog koda učenici mogu provjeriti na obrađenom primjeru igranja golfa u slučaju kada znamo da će biti sunčano i svježe vrijeme s visokom vlagom i jakim vjetrom. Za očekivati je da će algoritam vratiti *Ne*. Nakon što je funkcija za klasifikaciju dobro programirana može se krenuti na konkretne problemske zadatke.

Projekti

Jedna od možda najzanimljivijih primjena algoritma Naive Bayes je klasifikacija tekstualnih dokumenata. Takva primjena je vidljiva posvuda na internetu. Najbolji primjer toga bi bilo razvrstavanje e-pošte u različite kategorije. Pogledajmo Google i njegovu uslugu, e-pošte gmail, postoje četiri osnovne podjele: primarne, društvene, promocijske i gnjavaško (spam) e-pošiljke. Očito da je, zbog obujma poslanih e-pošiljka i sigurnosnih razloga, nemoguće razvrstavati ih ručno. Računalo mora "pročitati" tu pošiljku i odrediti u koju kategoriju da ju svrsta. Rješavanjem ovakvog tipa zadatka se rješava cijela jedna kategorija zadataka, jer se isti algoritam može koristiti u razne svrhe s nekim manjim izmjenama.

Projekt: Rješavanje gnjavaška

Osnovna ideja projekta je da se korištenjem Naivnog Bayesa računalo nauči prepoznavati dali se kod dane e-pošiljke radi o gnjavašku ili o normalnoj pošiljci koja bi korisnika možda zanimala. Ovakav projekt je jednostavno provesti, jer bi gotovo svi učenici trebali imati pristup svojim elektroničkim pošiljkama te informaciju o tome dali se kod pošiljke radi o gnjavašku ili ne. Osim što su te informacije lako dostupe radi se o nečemu što je učenicima veoma blisko i nešto s čime se sreću svakog dana.

Cilj projekta

Cilj ovakvog projekta je višestruk i složen. Moglo bi se reći da je cilj projekta naučiti računala da prepoznavanju dali se kod elektroničke pošte radi o gnjavašku ili ne. Ipak ja bi kod ovog projekta naglasak stavio više na prikupljanje i obradu informacija nego na rješavanje klasifikacije elektroničkih pošiljka. Prikupljanje i obrada podataka kao i procjena iskoristivosti takvih podataka je najveći dio strojnog učenja. Sami algoritmi su odavno svi dobro implementirani u raznim programima.

Očekivana učenička postignuća

- Prikupljati potrebne podatke iz raznih izvora,

- Analizirati i pohraniti podatke u prikladan format,
- Prilagoditi algoritam Naivni Bayes prikupljenim podacima,
- Rješavati razne probleme koji bi mogli nastati pri provođenju učenja modela,
- Diskutirati projekt s vršnjacima i nastavnicima,
- Provjeravati točnost naučenog modela,
- Analizirati rezultate naučenog modela,
- Prezentirati sva postignuća u projektu.

Kroz ovaj projekt s učenicima bi se trebali ostvariti neka odgojno-obrazovna očekivanja iz [2]Nacionalong kurikuluma međurpredmetne teme učiti kako učiti. Premda se ti ciljevi ne ostvaruju u predviđenom kontekstu ipak smatram da postoji mogućnost transfera tih postignuća na učeničko ponašanje kao što sam naveo u uvodu u ovaj rad. Odgojno-obrazovna očekivanja koja bi se nastojala postići ovim putem i poklapaju se s gore navedenim očekivanim postignućima su:

- a.1.3./4./5. Učenik samostalno traži nove informacije iz različitih izvora, transformira ih u novo znanje i uspješno primjenjuje pri rješavanju problema.
- a.2.4./5. Učenik se koristi različitim strategijama učenja i samostalno ih primjenjuje pri ostvarivanju ciljeva učenja i rješavanju problema u svim područjima učenja.
- a.3.4./5. Učenik kreativno djeluje u različitim područjima učenja.
- a.4.4./5. Učenik samostalno kritički promišlja i vrednuje ideje.
- b.1.4./5. Učenik samostalno određuje ciljeve učenja, odabire pristup učenju te planira učenje.
- b.4.3./4./5. Učenik samovrednuje proces učenja i svoje rezultate, procjenjuje ostvareni napredak te na temelju toga planira buduće učenje.

Nešto više o pretraživanju teksta i klasifikaciji tekstualnih podataka voditelj projekta može naći na stranicama 180. - 184. u knjizi [18]*Machine learning*. Nadalje, ako se projekt uspije uspješno provesti onda se isti može i nadograditi. Kao što sam spomenuo, Google za svoju uslugu elektroničke pošte, gmail, ima opciju razvrstavanja elektroničke

Spol	Dob	Vrijeme	Broj bradavica	Tip	Površina	Rezultat terapije
1	35	12	5	1	100	0
1	29	7	5	1	96	1
1	50	8	1	3	132	0
1	32	11.75	7	3	750	0
1	67	9.25	1	1	42	0
1	41	8	2	2	20	1

Tablica 2.7: Dio baze podataka krioterapije bradavica.

pošte u dodatne klase kao što je primarni pretinac, društveni pretinac i promocijski. Nadogradnja gore navedenog projekta bi mogla biti pokušaj ostvarivanje dodatne klasifikacije elektroničke pošte.

Možda najveći problem koji se javlja kod ovakvog projekta je činjenica da se elektronička pošta javlja u raznim jezicima. Za svaki jezik se treba stvoriti skup riječi po kojem bi se prepoznavao tekst. Dodatni problem svakako predstavljaju padeži koji riječi mijenjaju u nekoliko slova, ali ne i njihovo značenje. Na primjer, ako bi u nekom tekstu pisalo *"strojno učenje"* i *"strojnom učenju"* računalo ne bi znalo da se radi o istoj stvari. Na neki način bi se računali trebalo reći, ili ga naučiti, da se radi o istim stvarima zapisanim u drugim padežima. Samo to predstavlja dovoljno interesantan problem koji se treba riješiti.

Liječiti ili ne

Naivni Bayes ima svoju primjenu i u medicini. Može se primijeniti kao sustav koji će dati dijagnozu bolesnika, sugerirati način liječenja ili odrediti uspješnost liječenja pacijenta nekom danom metodom. Područje medicine je dobar primjer primjene algoritma, jer je njegova praktičnost očita, a rezultati odmah vidljivi i razumljivi.

U ovome projektu će se koristiti skup podataka [11]krioterapije bradavica. Skup podataka se sastoji od 90 pacijenata koji su prošli terapiju i o svakom pacijentu se ima 7 podataka. Odnosno ima 90 podataka na kojima se može učiti i svaki podatak ima 7 atributa, pogledati tablicu 2.7. Nakon naučenog modela, liječnici ga mogu koristiti pri donošenju odluke provođenja krioterapije.

Kao i kod prošlog projekta, naglasak se tu stavlja na pred procesiranje podataka. Podatke se treba analizirati, izabrati potrebne atribute, pretvoriti u oblik prikladan za programsku implementaciju, provesti učenje algoritma koristeći dane podatke, provjeriti preciznost dobivenog modela i argumentirati svoje postupke.

Ovaj projekt se može nadograditi tako da se pogleda jedan drugi skup podataka. Postoji skup podataka primjene [12]imunoterapije za rješavanje problema bradavica. Provodeći isti princip učenja nad ovom bazom podataka može se dobiti dodatni računalni model. U tome trenutku se dva dobivena modela mogu uspoređivati u pojedinim slučajevima pri određivanju najbolje terapije za pacijentov problem bradavica.

Provjera znanja

Neki od mogućih načina provjere znanja ovog gradiva uključuju:

- Opis principa rada algoritma naivnog Bayesa vlastitim riječima uz davanje primjera.
- Objašnjavanje osnovne pretpostavke nezavisnosti atributa i razloga zašto ona postoji.
- Ručno rješavanje problema klasifikacije primjenom naivnog Bayesa.
- Navođenje primjera problema gdje bi se algoritam mogao koristiti.

Poglavlje 3

Strojno učenje u svakodnevnom životu

U ovom poglavlju govorit ću o strojnom učenju u svakodnevnom životu. Cilj ovog poglavlja je da budućeg predavača, a i učenike, upozna sa nekim modernim primjenama strojnog učenja kao i osvještavanje sveprisutnosti strojnog učenja u današnjem životu. U doba kada su svi povezani sa svima i skoro svi imaju pristup internetu važno je razumjeti što se događa u tom digitalnom svijetu. Što više znamo o stvarima koje se događaju oko nas to lakše se možemo prilagoditi promjenama koje se događaju i koje dolaze.

3.1 Informacije kao roba

U prošlim poglavljima su se stalno koristili skupovi podataka za učenje računala. Očito je da bez takvih skupova podataka nebi bilo moguće podučavati računala niti rješavati dane probleme. Prikupljanje dovoljnog broja potrebnih podataka, i njihova interpretacija, predstavlja najveći problem pri korištenju strojnog učenja kod rješavanja problema. U zadnjih nekoliko godina, procvatom interneta i ponajviše društvenih medija, prikupljanje informacija je postalo znatno lakše. Količina informacija koje društveni mediji prikupljaju je zapanjujuća. O tome govori činjenica da je Facebook sagradio [16]skladište podataka imenom *Hive* koje je sposobno držati 300 petabyte-a podataka. Prikupljene podatke društveni mediji koriste za razne svrhe. Neke od njih su: prepoznavanje lica, reklamiranje, manipuliranje izbora i sl.

Jedan od najpoznatijih skandala vezanih za prikupljanje informacija je svakako prikupljanje podataka sa strane Cambridge Analytica od društvenog medija Facebook. [10]Cambridge Analyitica je uspio prikupiti podatke približno 87 milijuna korisnika. Te podatke je iskoristio kako bi prodao korisne informacije predsjedničkoj kampanji Donald-a Trump-a. Informacije su bile prikupljene tako da se korisnicima Facebook-a nudila opcija ispunjavanja psihološkog upitnika koji bi na kraju ljudima rekao nešto o njihovim osobnostima. Naravno, prije, nego što se taj upitnik može ispunit, korisnik mora aplikaciji dati pristup

svom profilu.

Nakon što su informacije procurile što se dogodilo došlo je do promjena u sigurnosti informacija korisnika društvene mreže Facebook. Iz osobnog iskustva mogu reći da je, za vrijeme dok je Cambridge Analytica prikupljao podatke, Facebook bio preplavljen raznim upitnicima i kvizovima koju su hvatali ljude s zanimljivim naslovima poput: *Koji element ste vi?, Koji Nordijski bog ste vi?, Koja je vaša duhovna životinja?, Kada ćete i kako umrijeti?* i sl. Sve, naravno, samo ako toj aplikaciji date pristup svom profilu. Nakon ovog skandala takvi kvizovi i upitnici su, čini se, potpuno nestali.

Slika 3.1: Slika kviza: *Koji Nordijski bog si ti?*

Naravno, ne mora svo prikupljanje informacija biti toliko kontraverzno i buditi skepticizam ljudi u moralnost korporacija društvenih medija. Jedan primjer dobrog poslovnog modela može se vidjeti kod internet igre imenom Covet Fashion. Cilj igre je dosta jednostavan. Nakon što napravite korisnički račun možete izabrati nekog modela te ga obući u razne haljine i kombinacije, ukratko, stvarate dizajn na predodređenu temu. Nakon što ste napravili dizajn drugi ljudi vaš dizajn mogu ocjenjivati. Ovisno o tome koliko dobru ocjenu dobijete to više dijamanata, ili neke druge virtualne valute, osvojite. S tim valutom onda možete kupovati nove odjevne predmete za što bolje dizajne. U međuvremenu igra je napredovala i dopustila ljudima da se udruže u dizajnerske kuće gdje kao kolektiv dobivaju više povlastica. Naravno, ako povežete svoj korisnički račun s Facebook profilom također dobijete neke povlastice.

Prije, nego što nastavim o ovoj igri, samo ću reći da, za ono što ću dalje navesti, nemam nikakve dokaze. To predstavlja moja razmišljanja i mogućnosti što se s ovakvom igrom može napraviti, a ne što se radi. Razmotrimo sada na koji način strojno učenje može primijeniti u toj igri. Vlasnici te igre imaju pristup raznim informacijama. Neke od tih informacija su: dob, spol, lokacija, obrazovanje i sl. Osim tih informacija također imaju

Slika 3.2: Covet Fashion

informaciju što se kojoj osobi sviđa. S tim informacijama, i raznim algoritmima strojnog učenja, mogu složiti demografsku sliku poželjnih odjevnih kombinacija. Na primjer, kroz svoju široku populaciju igrača mogu saznati da je trend žena u dobi između 20 i 25 godina tamnije kombinacije boja. Takvu informaciju mogu prodavati trgovačkim kućama. Osim što mogu prodavati informacije o trendovima, također mogu prodavati ciljano reklamiranje proizvoda modnih kuća svojim korisnicima, jer znaju njihov ukus. Štoviše, jer znaju kakve kombinacije većina ljudi voli, modnim kućama mogu i prodavati dizajn ideje.

Na kraju ovog pod poglavlja, samo ću navesti važnost da se učenike osvijesti na koji način se danas internet koristi i zašto se i kako informacije prikupljaju. Informacije su postale roba s kojom se trguje, samo se treba znati kome prodati kakve informacije. Danas je svima, više manje, jasno da se preko interneta prikupljaju podaci, ali smatram da većina zapravo ne razumije na koji način. Kroz strojno učenje i primjere iz života učenicima se može dati šira slika priče tako da budu svjesni kome i kada davati kakve informacije. Smatram da će takvo znanje biti još više potrebno sada kada se govori o internetu stvari, jer će prikupljanje informacija doseći još jedan vrhunac.

3.2 Umjetna inteligencija

Strojno učenje je jedna od grana umjetne inteligencije. Ova grana istraživanja je, unatoč velikim napredcima ostvarenih u zadnjih 20 godina, i dalje u povojima. Ipak, smatram da će za dvadeset ili trideset godina postati popularna i zaživit kao dio svakodnevnice. Naznake početka doba umjetne inteligencije su tu sa projektima poput samoupravljanih auta, Siri, Cortana i humanoidni roboti. Treba naglasiti da umjetna inteligencija nije strogo disciplina računalne znanosti.

[4] *Umjetna inteligencija se često prikazuje kao umijeće da se računala učine inteligentnima, što ovdje znači da ih se učini sposobnima da sami ili uz pomoć operatora izvršavaju zadatke za koje je potrebna inteligencija. Prema tomu, umjetna inteligencija je grana informatike, a ukoliko je njezin cilj ostvariti raznovrsne primjene oslanjajući se na čitav znanstveni korpus za koji prosudi da je koristan, ona je tehnologija.*

Ipak od početka , a i danas, u službenim raspravama za neupućene, kao i u napisima nekih stručnjaka upućenima kolegama, umjetna inteligencija se potvrđuje kao znanost. Njezin predmet? Inteligencija, spoznaja, sistemi obrade podataka - mišljena su podijeljena. U svakom Slučaju umjetna inteligencija zahtjeva svoje mjesto među spoznajnim znanostima, kadšto i prvo.

Na kraju, sve važnije mjesto što ga umjetna inteligencija u svome poimanju inteligencije i u simulacijama koje ostvaruje pridodaje "spoznajama" navelo je neke da istaknu razliku između "epistemičke" i "heurističke" komponente inteligencije. Prva, pogodna za gotovo logičko opisivanje, neovisna je o drugoj koja se odnosi samo na tehnike provedbe epistemičkih sadržaja. Tako se zadaća umjetne inteligencije približava jednom od glavnih zadataka filozofije, a to je znati opisati i razvrstati ljudske spoznaje.

Iz gornjeg teksta da se zaključiti da je područje umjetne inteligencije multidisciplinarno, a ne isključivo područje računalne znanosti. Zbog popularne kulture ljudi umjetnu inteligenciju poistovjećuju sa robotima ili pak nekim zlokobnim super računalima koja su stekla svijest. Za vrijeme događaja, *Dan za znanost*, koje se odvijalo Gimnaziji "Fran Galović", Koprivnica, u kojem sam držao predavanje o strojnom učenju i umjetnoj inteligenciji, jedan učenik me pitao dali se mi bavimo robotima odnosno imamo li robote. U tome trenutku sam s tim učenikom ušao u diskusiju što to uistinu je robot, što je umjetna inteligencija i može li uopće postojati takva stvar kao umjetna inteligencija. Po Hubert-u Dreyfus-u tako nešto nije uistinu moguće jer [9] *on tvrdi da "pravila ne sadrže pravila vlastite primjene". Odnosno pravila ne sadrže sve nužne informacije o kontekstu u kojem bi trebala biti primijenjena jer bi to zahtijevalo daljnja pravila koja bi objašnjavala ta pravila itd., ad infinum.*

[9] *Uzmimo za primjer sljedeći navod iz novina "Arizona Daily Star" od 31. svibnja 1986. "'Neiskusni vozač autobusa suspendiran je jer nije postupio kako nalaže slučaj nužde, kada je mlada djevojka doživjela srčani udar u autobusu. On je ustvari samo slijedio pretjerano stroga pravila koja priječe vozaču da napusti svoju rutu bez dopuštenja - izjavio je jučer predstavnik sindikata.*

Konvencionalno programiran program je poput spomenutog vozača autobusa; može slijediti pravila samo u jednom smjeru - ne može prepoznati kada kontekst zahtijeva da ih prekrši ili sijedi na drugi način.

Kada bi ste odlučili stvoriti elektronskog vozača autobusa, mogli bi ste ubaciti sljedeću naredbu: "Uvijek se pridržavaj rute osim ako neki putnik doživi srčani udar u kojem slučaju vozi u najbližu bolnicu." Takva naredba bi riješila incident poput ovoga u Arizoni. No, pun raspon intelektualnih mogućnosti ne bi se mogao anticipirati bez beskonačnog niza podataka.

3.3 Primjene strojnog učenja

Kroz cijeli rad sam navodio neke primjere primjene strojnog učenja u svakodnevnom životu. Ipak, smatram da se treba imati šira slika toga kako se strojno učenje koristi da bi se dala što bolja slika učenicima te ih se motiviralo na rad. Ovo su neki od primjera primjena.

Ekspertni sustavi

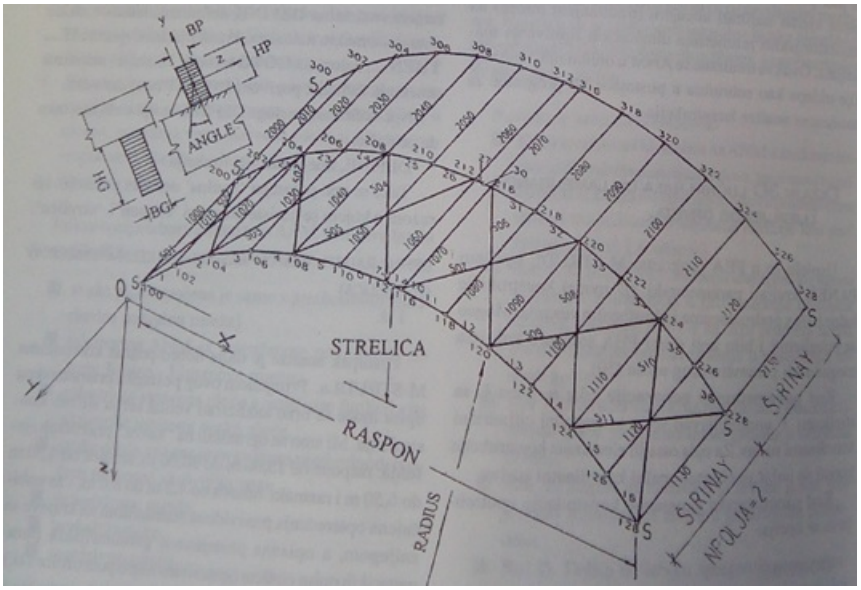
Ekspertni sustavi se danas koriste u raznim znanstvenim granama. Njihov cilj je zamijeniti ekspertno mišljenje stručnjaka tako da uče iz baze podataka po principima koje su zadali sami stručnjaci. [23]*Naime, znanje se u svim područjima ljudske djelatnosti širi vrtoglavom brzinom i, zato kompjutor može biti bolji od grupe eksperata. Samo za ilustraciju navedimo podatak da je procjena za područje kemije devedesetih godina* 20. *stoljeća bila da bi trebalo pročitati* 600000 *radova godišnje da se ostane dobro informiran! Dakako, to je nemoguće. "Spas" leži u ekspertnim sustavima kreiranim od tima eksperata, jer pojedinačni eksperti već znaju premalo!*

[23]*Prvi ekspertni sustav pod nazivom DENDRAL razvijen je na Stanfordskom univerzitetu* 1965. *godine za određivanje molekularne strukture kemijskih sastojaka. Desetak godina kasnije razvijen je prvi medicinski ekspertni sustav, tzv. MYCIN, koji je služio za postavljanje medicinske dijagnoze.*

Kao što se vidi iz primjera, ekspertni sustavi mogu biti veoma korisni. Premda izazivaju skepticizam kod ljudi, pogotovo kada je u pitanju nešto kao zdravlje, ipak se ukazalo da su precizniji od ljudskih stručnjaka. Danas se ti sustavi mogu koristiti za donošenje odluka ili, kod delikatnijih stvari, kao savjetnici koji informacije mogu obraditi brže od stručnjaka.

Modeliranje

Jedna od možda najinteresantnijih primjena strojnog učenja je modeliranje predmeta i struktura bez velike potrebe arhitekata. Jedan od takvih primjera je korištenje neuralne mreže kod projektiranja 3D lučnih hala od lameliranog drveta, kao što se može vidjeti na 3.3. Nakon što su se prikupili podaci raznih primjera sličnih lučnih konstrukcija koristila se neuralna mreža kako bi ju se naučilo da sama dizajnira slične objekte ovisno o unesenim parametrima. [7] *Nakon učenja u trajanju cca 3 minute, ljuska je pronašla optimalno rješenje, poređeno s prethodno nasumice odabranim setom parametara. Mreža je načinila* 22800 *ciklusa u* 949 *epoha učenja i posljednjom minimalnom greškom od* $0,0059$. *Grafički i numerički podaci pokazuju da je ljuska shvatila sve finese dizajna hale, te da dimenzije koje ona predviđa za neviđene zadane parametre odgovaraju konvencionalnim dimenzijama.*



Slika 3.3: Način numeracije čvorova i konačnih elemenata parametarskog upisa opisa konstrukcije 3D lučnehale za FEA programom M-STRUDL

Za razliku od gornjeg primjera, primjer kojeg ću sad navesti ne koristi neuronske mreže, već genski algoritam. Genski algoritam radi na principu da se nekom računalnom objektu daju genotipe te sposobnost stvaranja potomaka. Nakon toga se zada neka poželjna svojstva te se pusti da ti objekti evolviraju. Samo oni objekti koji su bliže poželjnim svojstvima stvaraju potomke i prenose svoje genotipe. Ideja je da će nakon određenog broja generacija ostati najprikladniji objekt koji odgovara poželjnim svojstvima.

[5]Peter Bentley, u knjizi *Digitalna biologija* navodi primjer svoga stolića za kavu. U svom programu on je stvorio digitalni svijet u kojem mogu postojati čvrsti objekti. Uvjeti koji su postavljeni na objekte su bili da se ne smiju prevrtati, da čestice koje lete na njega moraju ostati na određenoj visini te da ima što manju težinu. Što su manje ti objekti odgovarali tim uvjetima to je bila slabija šansa da svoje genotipe prenesu na sljedeći naraštaj i obratno. Evolucija objekta se mogla promatrati golim okom. Kako su prolazile generacije objekti su postajali sve ravniji i sve sličniji stolovima. Nakon 500 generacija konačni se model pospremio na disk te se stol izradio.

U nekoj školi koja posjeduje 3D printer, ovakav pristup osmišljanja predmeta bi bio dosta zanimljiv. Samo stvaranje umjetnog svijeta, pravila okoliša, objekta koji žive u tom svijetu te pravila prenošenja poželjnih atributa daje mogućnost pokretanja jednog iznimno kreativnog projekta koji bi mogao trajati generacijama.

Igre

Jedna primjena strojnog učenja koja u zadnje vrijeme postaje sve interesantnija je primjena u igrama. Odnosno, u stvaranju računalnog igrača koji bi u pojedinim igrama bio bolji profesionalaca. Taj trend se započeo s računalima koja igraju šah. Najpoznatija partija šaha bi svakako bila G. Kasparov protiv superračunala Deep Blue. [3] Godine 1997. Deep Blue pobjeđuje G. Kasparova u zadnjoj od 6 igara po standardnim pravilima. Nakon te igre Deep Blue se povlači i ostavlja gorak okus kod Kasparova koji je smatrao da je način na koji je Deep Blue igrao bio preljudski za stroj.

Kako su znanstvenici uspjeli napraviti stoj koji može pobijediti i velemajstora u šahu taj izazov više nije bio interesantan. Ubrzo se pojavila nova igra koja predstavlja znatno veći izazov, a ta igra je Go. Najveća razlika tih dvaju igara je ta što šah ima procijenjenih 10^{120} kombinacija dok Go ima oko 10^{170}. Ta igra predstavljala je znatno veći izazov, jer se Go ne može igrati metodom grube sile kao što je Deep Blue djelomično radio protiv Kasparova. Ipak, točno dvadeset godina nakon pobjede Deep Blue-a nad Kasparovom došao je i trenutak da računalo pobjedi velemajstora u igri Go. [19]Godine 2017. AlphaGo, računalna umjetna inteligencija Google-a, pobijedila je kineskog velemajstora Go-a, Ke Jie.

Slika 3.4: Ke Jie protiv AlphaGo, 2017.

Nakon ove pobjeda netko bi pomislio da se više nema što pobijediti, jer je Go jedna od najkompleksnijih igara koje trenutačno postoje, Ipak to bi bila samo varka. Fokus se sada prebacuje na online igre kao što je Dota2. Ta igra je trenutačno jedna od vodećih esport igara u svijetu. Nagrade za pobjedu turnira u toj igri doseže i vrijednosti od [22]40

milijuna dolara. Taj izazov je prihvatio tim OpenAI koji nastoji stvoriti računalo koje će biti sposobno pobijediti profesionalce u toj igri. Sama igra se sastoji od dva tima od po pet igrača koji na mapi s različitim herojima moraju pobijediti protivnički tim tako da im unište bazu. Već 2017. godine OpenAI je naučilo računalo da igra jedan na jedan te su uspjeli s tim računalom pobijediti profesionalce. Nakon pobjede su najavili da će do sljedeće godine naučiti svoja računala da igraju pet protiv pet te najavili svoju pobjedu protiv profesionalaca. Bez sumnje će nadolazeći turnir Dota2 igre biti zanimljiv.

Naravno, ne mora sve uvijek biti na samom rubu mogućnosti i gurati granice mogućnosti strojnog učenja. Neki od jednostavnijih primjera mogu se vidjeti kod YouTube korisnika pod pseudonimom Code Bullet. Taj čovjek stvara video sadržaje u kojima opisuje kako je naučio računala da igraju igre poput:[8] šaha, Worlds Hardest Game, Google dinosaur i Minesweper. Za školske primjere ovakve igre bi bile interesantne kao projekti, štoviše, zato što su neke od tih igara trenutačno popularne među učenicima.

3.4 Razredni projekt

U ovom poglavlju ću proći kroz dva razredna projekta koja bi se mogla provesti, ne samo u sklopu nastave strojnog učenja, nego i u sklopu nastave informatike. Jedan projekt je lakši, a drugi nešto zahtjevniji. U prvom projektu će se raditi o stvaranju računala koje igra Hexapawn igru, a u drugom križić kružić. Dobra stvar kod ovih projekata je što se mogu lako demonstrirati i široj publici nakon što se završe.

Hexapawn

Dosada sam spominjao samo razne algoritme učenja i kako se primjenjuju, ali potrebo je nešto konkretno. Zato bi ja predložio sljedeći projekt izrade računala koje uči na greškama i postaje sve bolje u igranju određene igre što više partija odigra. Ova [14]igra, nazvana hexapawn, je opisana u časopisu *Scientific American* 1962. godine. Radi se o igri za koju je potrebna šahovnica dimenzija 3x3, šest pijuna od kojih tri bijela i tri crna, 24 kutijice i četiri raznobojna predmeta u 24 primjerka svaki.

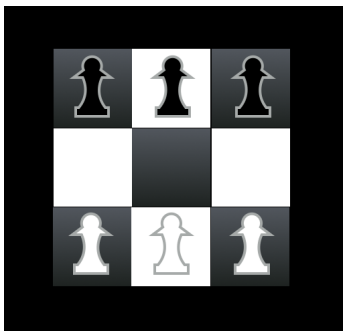
Pravila igre su sljedeća:

- Svaka grupa pijuna postavljena je na svojoj strani šahovnice u jednom redu, slika 3.5.
- Pijun se može kretati jedno polje naprijed ako je to mjesto slobodno.
- Pijun može jesti protivničkog pijuna, samo ako mu se nalazi lijevo ili desno na dijagonalnom polju.

- Pojedeni pijuni su maknuti iz igre.

Igrač pobjeđuje ako:

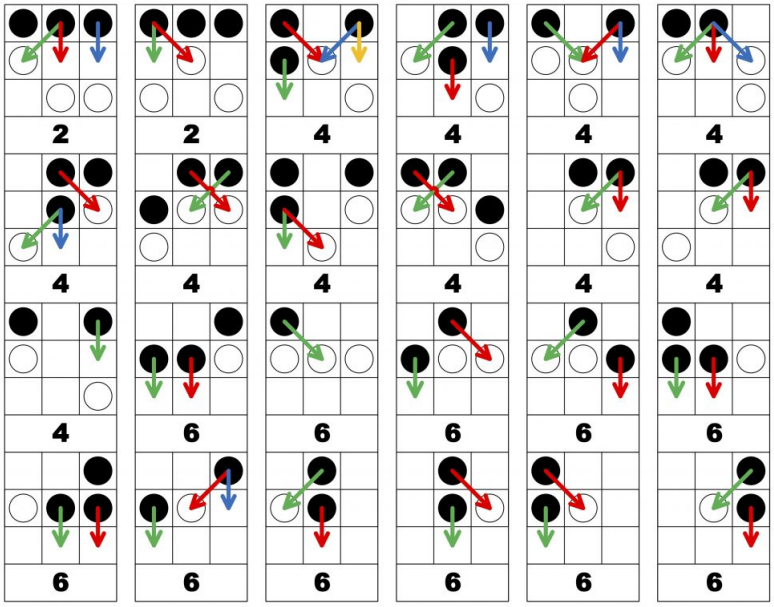
- Dovede svog pijuna na jedno od početnih polja protivničkih pijuna.
- Pojede sve protivničke pijune.
- Protivnik ne može pomaknuti nijednog svog pijuna.



Slika 3.5: Početna pozicija figurica Hexapawn igre.

Izrada ovakve šahovnice nije pretjerano teška i može se napraviti veoma jednostavno u jednom školskom satu, štoviše, učenici i mogu odigrati nekoliko partija. Zadatak pred koji se učenike može staviti jest da otkriju koliko jedinstvenih pozicija, ne računajući simetrične pozicije, igra može poprimiti. Uz malo igranja i analiziranja učenici bi trebali doći do zaključka da ima, kao što se vidi na slici 3.6, 24 jedinstvene pozicije za igru.

Sa šahovnicom, pijunima i kutijicama imamo osnovne komponente računala koje će učiti. Nedostaje samo da napunimo memoriju računala s nečim, a tu u igru dolaze oni raznobojni predmeti. Ako pogledamo sliku 3.6 uočit ćemo da je svaki mogući korak označen sa strelicama. Svaka od strelica označi se različitom bojom. Na svaku kutijicu nalijepimo jednu od mogućih pozicija potom u tu kutijicu stavimo predmete koji odgovaraju bojama strelica na slici. Također treba napomenuti da svaka pozicija ispod svoje slike ima brojeve 2, 4 ili 6. Ti brojevi označavaju redni broj poteza, odnosnu drugi, četvrti i šesti potez. Iz toga se da zaključiti da je namijenjeno da računalo uvijek ide drugo. Sada, kada se ima spremno računalo i igru, treba se odrediti algoritam po kojem će računalo učiti kako pobjeđivati. Jedna odlična stvar kod ove igre i ovog računala što je njegov princip učenja igre Hexapawn toliko intuitivan da se ovaj primjer može pokazati i učenicima osnovne škole. Pravila učenja, odnosno algoritam za učenje, za ovo računalo su sljedeća:



Slika 3.6: Sve moguće jedinstvene pozicije figurica bez simetrija.

- Provjeri koja je pozicija figurica na ploči.
- Nađi kutijicu koja ima odgovarajuću sliku pozicije te iz nje nasumično izvuci jedan predmet.
- Ako je izvučeni predmet doveo do pobjede vrati ga u kutijicu.
- Ako je izvučeni predmet doveo do gubitka on se izbacuje iz igre.
- Ako je kutijica prazna onda računalo odustaje i predmet iz prošle kutijice, koji ga je doveo do ove pozicije, se izbacuje iz igre.
- Ako je računalo pobijedilo, izgubilo ili odustalo svi ne izbačeni predmeti se vraćaju natrag u svoje kutijice.
- Vrati se na početni korak.

Kroz igru je vrlo jednostavno shvatiti princip na koji računalo uči. Štoviše može se pratiti napredak učenja tog računala, jer će sve manje i manje griješiti. Sveukupno će računalo 11 puta izgubiti igru prije nego što postane nepobjedivo. Koncept učenja kojeg primjenjuje ovo računalo je dosta intuitivno, ako me ovaj predmet doveo do gubitka izbacim ga iz kutijice i više nikad neću ponoviti taj potez.

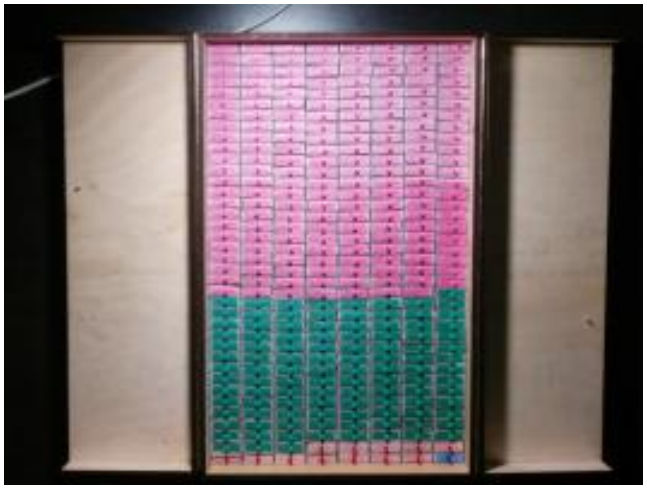
Slika 3.7: KiKi računalo uči igrati Hexapawn uz pomoć učenika gimazije Fran Galović.

Princip učenja ovog računala sam proveo s učenicima gimnazije Fran Galović, slika 3.7, u Koprivnici gdje sam držao predavanje na temu umjetne inteligencije. Računalo se sastojalo od papirnih kutijica u kojima su se nalazili raznobojni KiKi bomboni. Prikladno, računalo se zvalo KiKi računalo. Učenici su bili veoma zainteresirani za cijelu temu i bili su dosta aktivni. Pred kraj samog predavanja, nakon demonstracije KiKi računala učenike sam stavio pred sljedeća dva problema:

- Kako bi sagradili računalo da uzima u obzir pat situaciju?
- Kako bi sagradili računalo uzimajući u obzir da može ići prvo ili drugo?

Nije trebalo dugo čekati da da učenici osmisle koncept računala koje može raditi i gore navedene stvari. Učenici Danko Delimar, Oskar Marić, Lucija Đaković i Doris Hegedušić dali su sljedeća rješenja.

Za problem pat situacije postoje dva različita pristupa. Jedan pristup je bio da se dana kutijica podijeli u dva manja pretinca. Jedan za pobjedu, a drugi za pat situaciju. Algoritam bi se tako promijenio da se prvo gleda u kutijicu za pobjedu, ako u njoj ima predmeta onda se bira iz tog pretinca ako nema onda se bira iz pretinca za pat. Ako se dogodi pat situacija onda se objekt seli iz pretinca pobjede u pretinac za pat. Drugi pristup je bio napraviti dodatne 24 kutijice u kojima će se pamtiti isključivo pat situacije, a originalne kutijice pamte pobjede. Ako u skupu kutijica za pobjedu nema nikakvih predmeta onda se gleda u kutijice za pat. Ako predmet izvučen dovodi do pat situacije onda se on vadi iz kutijica za pobjedu, ako dođe do pobjede onda se taj predmet vadi iz kutijice za pat, a ako se izgubi onda se miče iz obje kutijice.



Slika 3.8: Računalo MENACE.

Za problem, kad računalo može ići prvo ili drugo, našlo se jedno rješenje, a to je bilo dodatne kutijice koje bi predstavljale slučaj kad računalo ide prvo s analognim pravilima učenja kao i kod originalnog problema.

Križić kružić

Jedno malo zanimljivije računalo bi svakako bilo [14]MENACE (Matchbox Educable Naughts And Crosses Engine), slika 3.8, koje se koristi paketićima šibica kako bi oponašao igranje popularne igre križić kružić. To računalo se sastoji od 304 paketića šibica. Svaki paketić predstavlja jedno moguće stanje igre ne računajući simetrije. U sebi kutije sadrže obojene perle za svaki mogući potez. Princip učenja je u potpunosti isti kao i kod igre Hexapawn, samo što ovom računalu treba znatno više igara kako bi naučilo pobjeđivati. Ovo bi bio jedan veći projekt u kojem bi mogla sudjelovati cijela škola koja bi onda mogla taj svoj stroj prezentirati na raznim školskim događanjima. Paralelno stvaranju fizičkog računala može se napisati i kod u nekom programskom jeziku koji bi simulirao to učenje. Time bi se moglo odrediti koliko igara je potrebno da računalo nauči. Štoviše može se usporediti dali računalo uči brže kad igra samo protiv sebe ili kad igra protiv ljudskih igrača. Za pretpostaviti je da će brže učiti ako igra samo protiv sebe, jer u jednoj igri izgubi i pobjedi.

Nadalje, kako bi računalo postiglo što bolju strategiju može se modificirati izvorni algoritam da računalo preferira one izbore koji dovode do pobjede, a izbjegava one koji vode do pat situacije. Ovaj projekt se može nadograđivati kroz više generacija učenika što ga čini odličnim školskim projektom.

Bibliografija

[1] *Multiple Linear Regression (solutions to exercises*, (3.09.2018.), https://02402.compute.dtu.dk/filemanager/enotes/02402/files/solutions-chapter6.pdf.

[2] *Nacionalni kurikulum međupredmetne teme Učiti kako učiti*, (3.09.2018), https://mzo.hr/sites/default/files/migrated/uciti_kako_uciti_prije_strucne_rasprave.pdf.

[3] M. R. Anderson, *Twenty years on from Deep Blue vs Kasparov: How a chess match started the big data revolution*, (18.08.2018.), http://theconversation.com/twenty-years-on-from-deep-blue-vs-kasparov-how-a-chess-match-started-the-b

[4] D. Andler, *Uloga umjetne inteligencije u proučavanju spoznaje*, Treći program hrvatskog radija (1990), br. 28, 122–133.

[5] P. Bentley, *Digitalna biologija*, Izvori, Zagreb, 2004.

[6] C.M. Bishop, *Pattern recognition and machine learning*, Springer Science+Business Media, LLC, 2006.

[7] J. Božičević, *Inteligentno vođenje i inteligentni sustavi*, Hrvatsko društvo za sustave, Zagreb, 1998.

[8] Code Bullet, *AIigre*, (18.08.2018.), https://www.youtube.com/channel/UC0e3QhIYukixgh5VVpKHH9Q/featured.

[9] H. Collins, *Hoće li strojevi ikada misliti?*, Treći program hrvatskog radija (1995), br. 48, 5–10.

[10] N. Confessore, *Cambridge Analytica and Facebook:The Scandal and the Fallout So Far*, (16.08.2018.), https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html.

[11] P. Layegh F. Khozeimeh, *Cryotherapy Dataset Data Set*, (1.09.2018.), https://archive.ics.uci.edu/ml/datasets/Cryotherapy+Dataset+.

[12] ———, *Immunotherapy Dataset Data Set*, (1.09.2018.), https://archive.ics.uci.edu/ml/datasets/Immunotherapy+Dataset.

[13] R. A. Fisher, *The use of multiple measurements in taxonomic problems*, Anual Eugenics (2018), br. 2, 178–188, https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1469-1809.1936.tb02137.x.

[14] M. Gardner, *Mathematical games*, Scientific American **206** (March 1962), br. 3, 138–144.

[15] A. Courville I. Goodfellow, Y. Bengio, *Deep Learning*, MIT Press, 2016.

[16] N. Bronsan J. Weiner, *Facebook's Top Open Data Problems*, (16.08.2018.), https://research.fb.com/facebook-s-top-open-data-problems/.

[17] Mayur Kulkarni, *Decision Trees for Classification: A Machine Learning Algorithm*, (23.07.2018.), https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html.

[18] Tom M. Mitchell, *Machine Learning*, McGraw-Hill Science/Engineering/Math, 1997.

[19] P. Mozur, *Google's AlphaGo Defeats Chainese Go Master in Win for A.I.*, (18.08.2018.), https://www.nytimes.com/2017/05/23/business/google-deepmind-alphago-go-champion-defeat.html.

[20] R. Ng, *Linear Regression with Multiple Variables*, (2.09.2018.), https://www.ritchieng.com/multi-variable-linear-regression/.

[21] PennState Eberly Collage of Science, *STAT 501*, (28.07.2018.), https://onlinecourses.science.psu.edu/stat501/node/252/.

[22] OpenAI, *OpenAI Five*, (18.08.2018.), https://blog.openai.com/openai-five/.

[23] P. Zarevski, *Struktura i priroda inteligencije*, "Naklada Slap", 2000.

Sažetak

Svrha ovog diplomskog rada je stvaranje osnova za uvođenje elemenata strojnog učenja u srednjoškolsko obrazovanje. U današnje doba strojno učenje se univerzalno koristi u gotovo svim sferama života. Od reklamiranja, prepoznavanja glasa, procjene vrijednosti nekretnina, analize podataka pa do određivanja dijagnoze pacijenta i arhitekture. Zbog naglog razvoja interneta i tehnologije kao i najave interneta stvari važno je učenike upoznati sa modernim primjenama računala. U današnjem obrazovanju se naglasak stavlja na cjeloživotno učenje i na vještinu učenja kako učiti da se ostvari cjeloživotno učenje. Strojno učenje u sebi sadrži osnovne principe učenja i zbog toga je dobro za razvijanje vještine učenja kako učiti. Ono također predstavlja jedan multidisciplinaran pristup koji u sebi koristi i objedinjuje spoznaje psihologije, matematike i računalnih znanosti. Strojno učenje može služiti kao dobar primjer primjene apstraktnog matematičkog znanja na stvarne probleme. Zbog svih navedenih razloga, smatram da se strojno učenje treba uvesti kao dio nastave u školama.

U ovom radu sam naveo osnovnu podjelu strojnog učenja na nadzirano, nenadzirano u podržano učenje. Naveo sam neke od algoritama koji se koriste u strojnom učenju i detaljnije obradio algoritme: stablo odlučivanja, linearna regresija i naivni Bayes. Za svaki navedeni algoritam dao sam i jedno moguće rješenje za njegovu implementaciju u programskom jeziku Python. Pred kraj rada sam naveo primjene strojnog učenja u svakodnevnici kao i neke događaje iz popularne kulture vezane na tu temu.

Summary

The purpose of this master's thesis is creating a foundation for the introduction of some elements of machine learning into secondary education. In today's age, machine learning is universally used in almost all aspects of life. From advertisement, voice recognition, value estimate of real estate, data analysis to determining patient diagnosis and architecture. Because of the rapid development of internet and technology and the announcement of internet of things it is important to familiarize students with the modern use of computers. In today's education there's an emphasis on lifelong learning and the skill of learning how to learn to achieve lifelong learning. Machine learning encompasses basic principles of learning within itself and therefor is a good way to develop the skill of learning how to learn. It also represents a multidisciplinary approach which uses and encompasses the knowledge of psychology, mathematics and computer science. Machine learning can be used as a good example of applied abstract mathematical knowledge on real life problems. Because of everything mentioned, it is my opinion that machine learning should be introduced into classes at schools.

In this thesis I have made a basic separation of machine learning into supervised, unsupervised and reinforced learning. I have named some algorithms commonly used in machine learning and talked about: decision tree, linear regression and naive Bayes in more detail. For each above mentioned algorithm I gave one possible implementation in the programming language Python. At the end of the thesis I have given examples of the use of machine learning in everyday life as well as a few events from popular culture tied to the subject.

Životopis

Dana 15.01.1988. rođen u Tomislavgradu. Sa pet godina sam se preselio u Zagreb gdje sam i pohađao OŠ Mate Lovrak. Godine 2006. završavam smjer mehatroničara u Strojarskoj tehničkoj školi Faust Vraničić. Iste godine se upisujem na Fakultet strojarstva i brodogradnja kojeg pohađam sljedeće tri godine. Godine 2009. se upisujem na preddiplomski sveučilišni studij Matematika; smajer: nastavnički kojeg završavam 2014. godine. Nakon preddiplomskog studija se upisujem na diplomski sveučilišni studij Matematika i informatika; smjer: nastavnički.