DEO I

Osnove mašinskog učenja

POGLAVLJE 1

Pejzaž mašinskog učenja

Kada većina ljudi čuje za „mašinsko učenje", zamisli robota: vernog slugu ili smrtonosnog Terminatora, u zavisnosti od toga koga pitate. Ipak, mašinsko učenje nije samo futuristička fantazija; ono već postoji. U stvari, postoji već decenijama u određenim specijalizovanim aplikacijama, kao što je optičko prepoznavanje znakova (OCR). Ali prva aplikacija MU koja je postala sveprisutna i poboljšala živote stotine miliona ljudi, proširila se svetom još od devedesetih godina je *filtar neželjene pošte*. To nije baš samosvesni Skynet, ali tehnički gledano, svrstava se u kategoriju mašinskog učenja (a toliko ga je dobro savladao, da vam retko zatreba da određenu poruku e-pošte sami označavate kao neželjenu). Tome su sledile stotine aplikacija MU koje danas čine srž snage stotina proizvoda i funkcija koje redovno koristite, od boljih preporuka do glasovnog pretraživanja.

Gde započinje mašinsko učenje i gde ono prestaje? Šta tačno znači za mašinu da *nauči* nešto? Ako ja preuzmem kopiju Wikipedije, da li je time moj računar zaista nešto naučio? Da li je odjednom postao pametniji? U ovom poglavlju počećemo razjašnjenjem šta je to mašinsko učenje i zašto biste poželeli da ga koristite.

Zatim, pre nego što započnemo istraživanje kontinenta mašinskog učenja, proučićemo njegovu mapu da bismo saznali koja su najvažnija područja i najistaknutiji reperi: nadgledano (eng. *supervised*) i nenadgledano (eng. *unsupervised*) učenje, onlajn nasuprot paketnom učenju, učenje na konkretnim primerima i učenje na modelu. Potom ćemo pogledati tok tipičnog projekta MU, razmotriti glavne teškoće na koje možete naići i objasniti kako biste ispitali i precizno podesili jedan sistem mašinskog učenja.

Ovo poglavlje uvodi veći broj osnovnih koncepata (kao i žargon) koje bi trebalo da zna napamet svaki naučnik koji se bavi analizom podataka. To će biti pregled na visokom nivou (ovo je jedino poglavlje bez mnogo koda), sve prilično jednostavno, ali trebalo bi da se postarate da vam sve bude kristalno jasno pre nego što nastavite čitanje preostalog dela ove knjige. Zato uzmite šolju kafe i počnimo!

Ako već poznajete osnove mašinskog učenja, možete preći direktno na poglavlje 2. Ako ste u nedoumici, pre nego što nastavite, pokušajte da odgovorite na sva pitanja navedena na kraju ovog poglavlja.

Šta je mašinsko učenje?

Mašinsko učenje je nauka (i umetnost) programiranja računara tako da oni *uče iz podataka*.

Evo jedne nešto opštije definicije:

> [Mašinsko učenje je] oblast istraživanja koja računarima omogućava da uče bez izričitog programiranja.
>
> Arthur Samuel, 1959.

A sledeća bi bila više inženjerski orijentisana:

> Kaže se da računarski program uči na osnovu iskustva *I* datog posla *P*, uz meru performanse *E*, ako se performansa pri obavljanju posla *P*, merene sa *E*, poboljšavaju sa iskustvom *I*.
>
> Tom Mitchell, 1997.

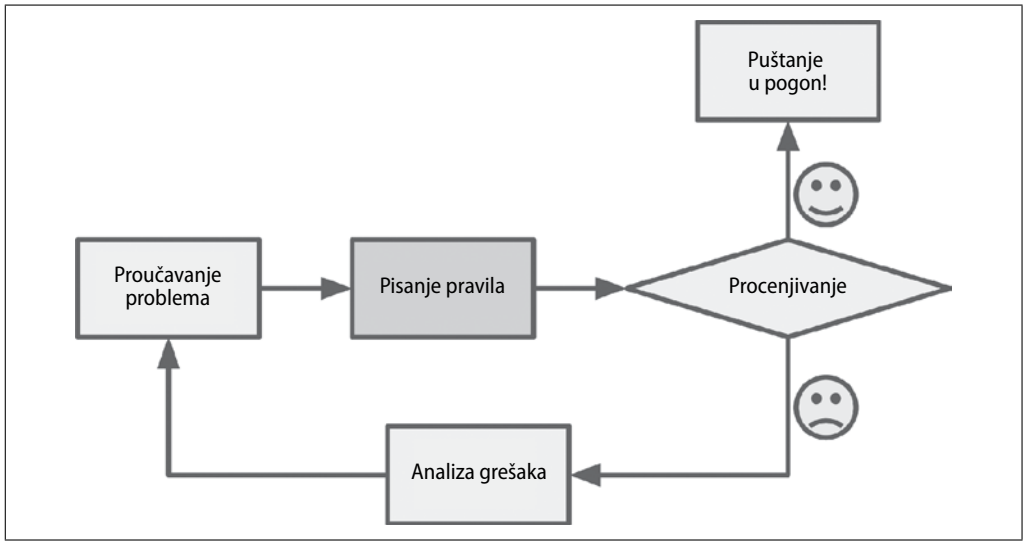
Vaš filtar za neželjenu poštu je program mašinskog učenja koji pomoću primera neželjene e-pošte (označene od strane korisnika) i primera željene e-pošte (eng. *ham*) može naučiti da izdvaja neželjenu poštu. Skup konkretnih primera na osnovu kojih sistem uči zove se trening skup, tj. skup za obuku (eng. *training set*). Svaki primer u trening skupu zove se primer za obuku (eng. *training instance*) ili uzorak (eng. *sample*). U navedenom slučaju, posao P je izdvajanje neželjenih iz skupa novih poruka e-pošte, iskustvo I su podaci za obuku, a meru performansi E tek treba definisati; na primer, za tu namenu možete iskoristiti procenat ispravno klasifikovanih poruka. Taj oblik mere performansi zove se tačnost i često se koristi u poslovima klasifikovanja nečega.

Ako samo preuzmete kopiju Wikipedije, vaš računar ima znatno veću količinu podataka, ali neće zbog toga ništa bolje obavljati nijedan posao. Dakle, preuzimanje kopije Wikipedije nije mašinsko učenje.

Zašto koristiti mašinsko učenje?

Razmotrimo kako biste napisali filtar za neželjene poruke primenom tradicionalnih tehnika programiranja (slika 1-1):

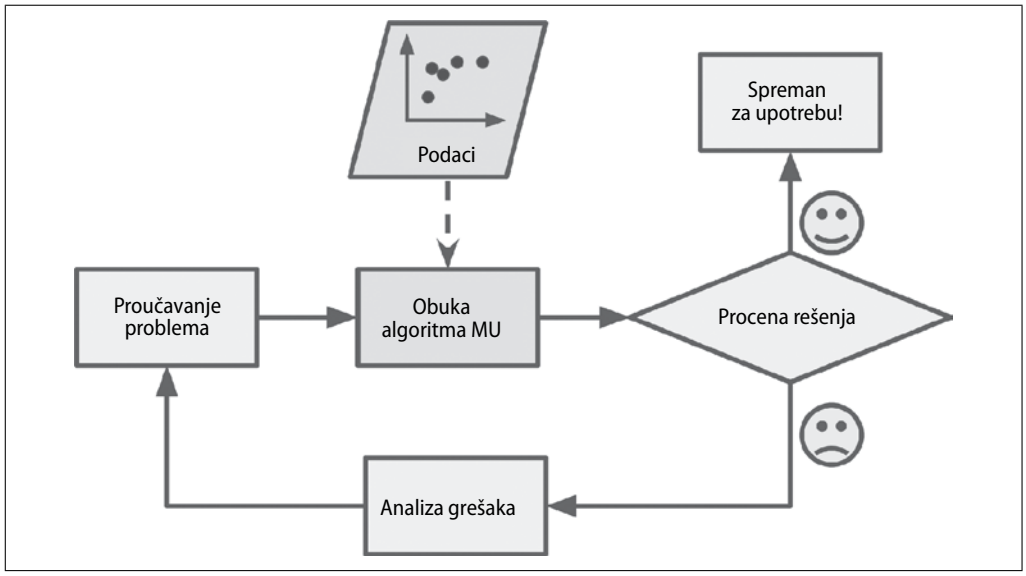
1. Prvo biste ispitali kako izgleda tipična neželjena poruka. Možda biste zapazili da se određene reči ili izrazi (kao što su „Vaš(a)", „kreditna kartica", „besplatno", i „fantastično") često pojavljuju u redu za predmet poruke. Možda biste uočili još nekoliko drugih šablona u imenu pošiljaoca, u telu ili drugim delovima poruke.
2. Napisali biste algoritam za prepoznavanje svakog šablona koji ste uočili, a vaš program bi obeležavao e-poštu kao neželjenu ako prepozna jedan ili više tih šablona.
3. Svoj program biste testirali i ponavljali korake 1 i 2 dok program ne postane dovoljno dobar za stavljanje u redovnu upotrebu.



Slika 1-1. Tradicionalni pristup

Pošto je problem težak, vaš program će verovatno postati dugačka lista složenih pravila koja će biti teška za održavanje.

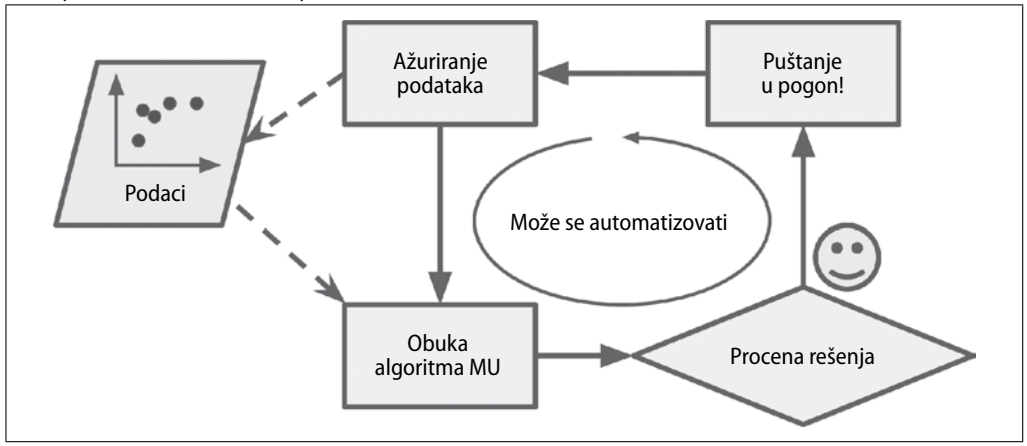
Nasuprot tome, filtar za neželjene poruke koji se zasniva na tehnikama mašinskog učenja automatski uči koje su reči i izrazi dobri pokazivači neželjenih poruka tako što otkriva neuobičajeno česte šablone reči u primerima neželjenih poruka koje upoređuje s primerima prihvatljivih poruka (slika 1-2). Program je znatno kraći, lakše se održava, a vrlo verovatno i tačnije radi.



Slika 1-2. Pristup mašinskog učenja

Šta ako pošiljaoci neželjenih poruka shvate da sve njihove poruke koje u naslovu sadrže „Vaš(a)" primalac blokira? Umesto toga mogu početi da pišu „Za Vas". Filtar za neželjene poruke napisan pomoću tradicionalnih programskih jezika bi onda trebalo dopuniti tako da izdvaja i poruke koje sadrže „Za Vas". Ako pošiljaoci neželjene e-pošte nastave da zaobilaze vaš filtar, moraćete da mu beskonačno dodajete nova pravila.

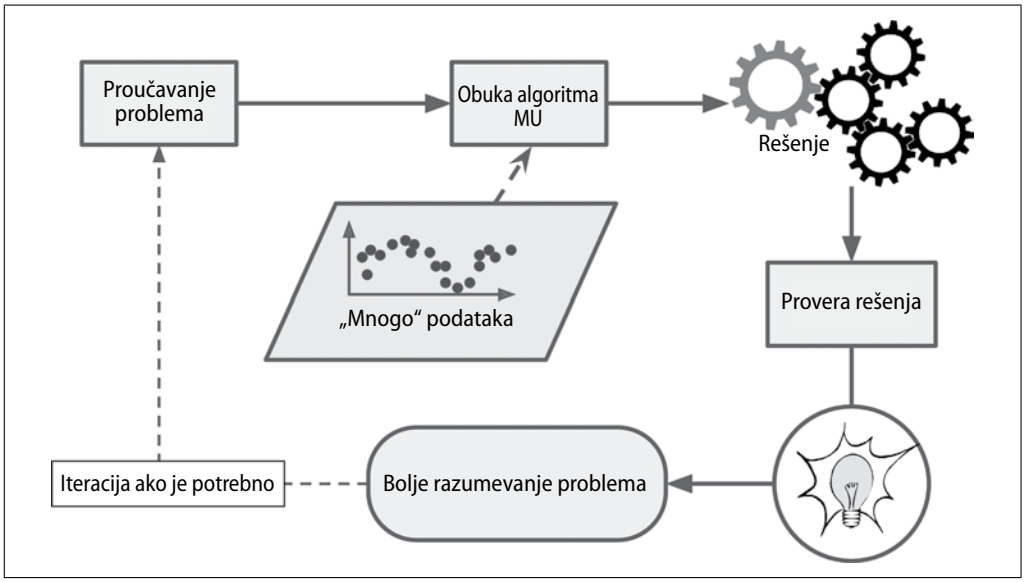
Nasuprot tome, filtar za neželjene poruke koji se zasniva na tehnikama mašinskog učenja automatski prepoznaje da je izraz „Za Vas" (4U u eng. izdanju ove knjige) postao neuobičajeno čest u neželjenim porukama koje su korisnici tako obeležili, pa zato počinje da ih sam izdvaja bez vaše intervencije (slika 1-3).



Slika 1-3. Automatsko prilagođavanje promeni

Još jedna oblast u kojoj mašinsko učenje briljira je kada imate probleme koji su ili previše složeni za tradicionalne pristupe, ili za njihovo rešavanje nema poznatog algoritma. Na primer, prepoznavanje govora. Recimo da želite da krenete od nečeg jednostavnog i napišete program koji je u stanju da razlikuje reči „on" i „ti". Verovatno biste zapazili da zamenica „ti" počinje zvukom više frekvencije („T"), pa biste u direktno u program ugradili algoritam koji meri jačinu zvukova više frekvencije, da biste razlikovao „on" od „ti" – ali jasno je da ta tehnika neće moći da se primeni na hiljade reči koje izgovaraju milioni različitih ljudi u bučnim okruženjima i na desetinama raznih jezika. Najbolje rešenje (barem danas) je da napišete algoritam koji sam uči i kojem pružite veliki broj snimaka primera izgovaranja svake reči.

I najzad, mašinsko učenje može pomoći i ljudima da uče (slika 1-4). Algoritmi za MU mogu se ispitivati radi utvrđivanja šta su oni naučili (mada za neke algoritme to može biti zapetljano). Na primer, pošto filtar za neželjene poruke obučite na dovoljno primera takvih poruka, lako ga možete ispitaiti da bi vam on pokazao listu reči i kombinacija reči za koje on „veruje" da su najbolji pokazivači neželjenih poruka. Ponekad će to otkriti neočekivane korelacije ili nove trendove, što će vas dovesti do boljeg razumevanja problema. Primena tehnika MU za analiziranje velikih količina podataka može pomoći za otkrivanje šablona koji nisu uočljivi na prvi pogled. To se zove *rudarenje podataka* (eng. *data mining*).



Slika 1-4. Mašinsko učenje može pomoći ljudima da uče

Ukratko, mašinsko učenje je odlično za:

- Probleme za koje postojeća rešenja zahtevaju mnogo finog podešavanja ili dugačke liste pravila: jedan algoritam mašinskog učenja često može da pojednostavi kôd i radi bolje nego tradicionalni pristup.
- Složene probleme za koje tradicionalni pristup ne pruža dobro rešenje: najbolje tehnike mašinskog učenja možda mogu pronaći rešenje.
- Promenljiva okruženja: sistem mašinskog učenja se može prilagođavati novim podacima.
- Sticanje uvida o složenim problemima i opsežnim količinama podataka.

Primeri primene

Pogledajmo neke konkretne primere zadataka u kojima se koristi mašinsko učenje, zajedno sa tehnikama:

Analiza slika proizvoda na proizvodnoj liniji radi njihove automatske klasifikacije
: Ovo je klasifikovanje slika, koje se uglavnom obavlja pomoću konvolucionih neuronskih mreža (CNN; poglavlje 14).

Otkrivanje tumora u skenovima mozga
: Ovo je semantičko segmentiranje, gde se svaki piksel klasifikuje (pošto želimo da odredimo tačno mesto i oblik tumora), uglavnom takođe pomoću CNN mreža.

Automatska klasifikacija vesti
Ovo je obrada govornog jezika (NLP), tačnije klasifikovanje teksta, što se može obavljati pomoću rekurentnih neuronskih mreža (RNN), CNN mreža ili transformera (poglavlje 16).

Automatsko obeležavanje uvredljivih komentara na diskusionim forumima
Ovo je takođe klasifikovanje teksta, pomoću istih NLP alatki.

Automatska izrada rezimea dugačkih dokumenata
Ovo je grana NLP-a koja se zove sažimanje teksta i takođe se radi pomoću istih alatki.

Izrada četbota ili ličnog asistenta
Ovo podrazumeva upotrebu mnogih NLP komponenata, među kojima je i razumevanje govornog jezika (NLU), kao i modula za odgovaranje na pitanja.

Predviđanje prihoda kompanije u narednoj godini na osnovu velikog broja podataka o njenim performansama u tekućoj
Ovo je izrada regresije (tj. predviđanja vrednosti), što se može obaviti pomoću proizvoljnog modela regresije, kao što je model linearne regresije ili model polinomijalne regresije (poglavlje 4), SVM regresije (pogavlje 5), regresije nasumične šume (pogavlje 7), ili pomoću veštačke neuronske mreže (pogavlje 10). Ako želite da uzmete u obzir i sekvence podataka o ranijim performansama, najbolje rešenje može biti upotreba RNN mreža, CNN mreža ili transformera (poglavlja 15 i 16).

Ugrađivanje mogućnosti da aplikacija reaguje na glasovne komande
Ovo je prepoznavanje govora, što zahteva obradu audio uzoraka: pošto su to dugačke i složene sekvence, obično se obrađuju pomoću RNN mreža, CNN mreža ili Transformera (poglavlja 15 i 16).

Otkrivanje prevara sa kreditnim karticama
Ovo je otkrivanje anomalija (poglavlje 9).

Raspodela klijenata u segmente na osnovu njihovih kupovina da biste za svaki segment razradili različitu marketinšku strategiju
Ovo je grupisanje podataka (poglavlje 9).

Predstavljanje složenog skupa podataka, sa velikim brojem dimenzija, u obliku jasnog i rečitog dijagrama
Ovo je vizualizacija podataka, što često podrazumeva upotrebu tehnika za smanjivanje broja dimenzija (poglavlje 8).

Preporučivanje proizvoda koji može biti zanimljiv klijentu na osnovu njegovih ranijih kupovina
Ovo je sistem za preporuke. Jedan od pristupa je da se podaci o ranijim kupovinama (i drugi podaci o klijentu) proslede veštačkoj neuronskoj mreži (pogavlje 10) da bi se iz nje dobila najverovatnija sledeća kupovina. Ta neuronska mreža uči obično na ranijim sekvencama kupovina svih klijenata.

Izrada inteligentnog bota za igricu

Ovo se često rešava pomoću forsiranog učenja (eng. *reinforcement learning*, RL); pogavlje 18, koje je grana mašinskog učenja koja uči agente (kao što su botovi) da biraju one akcije koje će tokom vremena maksimizovati njihove nagrade (npr. bot može dobiti nagradu kad god protivnički igrač izgubi određen broj poena koji mu produžavaju život u igri), unutar datog okruženja (kao što je igrica). Čuveni program AlphaGo, koji je pobedio svetskog šampiona u igri Go, napravljen je pomoću RL-a.

Ovaj spisak se može nastaviti, ali trebalo bi da vam pruži sliku neverovatne širine i složenosti poslova koji se mogu obavljati pomoću mašinskog učenja, kao uvid u tehnike koje biste koristili za svaki od tih poslova.

Vrste sistema mašinskog učenja

Postoji mnogo vrsta sistema mašinskog učenja, pa je korisno da ih razvrstamo u određene opšte kategorije na osnovu sledećih svojstava:

- Da li oni uče pod nadzorom čoveka ili bez toga (nadgledano, nenadgledano, polunadgledano i forsirano učenje)
- Da li mogu ili ne mogu da uče postepeno u hodu (onlajn nasuprot paketnom učenju)
- Da li rade tako što samo porede nove podatke s već poznatim podacima ili otkrivaju šablone u podacima za obuku i grade model na osnovu kojeg će zatim predviđati, vrlo slično načinu na koji rade naučnici (učenje na konkretnim primerima nasuprot učenju na modelu)

Ova svojstva nisu ekskluzivna; možete ih kombinovati na proizvoljne načine. Na primer, najsavremeniji filtar za neželjene poruke e-pošte može učiti u hodu pomoću modela duboke neuronske mreže koji je prethodno obučen na konkretnim primerima neželjenih i prihvatljivih poruka. To ga čini onlajn nadgledanim sistemom mašinskog učenja na modelu.

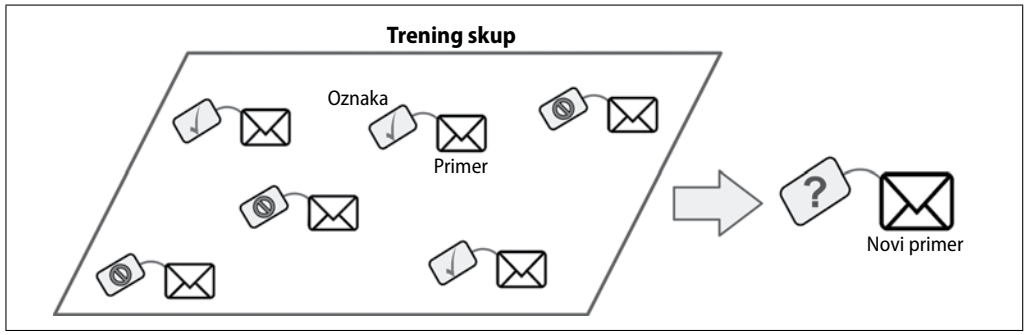
Razmotrimo malo detaljnije navedene mogućnosti.

Nadgledano/nenadgledano učenje

Sistemi mašinskog učenja mogu se razvrstavati na osnovu oblika i opsega nadgledanja koji im je potreban tokom postupka učenja. Postoje četiri glavne kategorije: nadgledano učenje, nenadgledano učenje, polunadgledano učenje i forsirano učenje.

Nadgledano učenje

Kod *nadgledanog učenja* (eng. *supervised learning*), trening skup koji prosleđujete algoritmu sadrži poželjna rešenja, koja se zovu *oznake* (eng. *labels*) (slika 1-5).



Slika 1-5. Trening skup dopunjen oznakama za klasifikovanje neželjenih poruka (primer nadgledanog učenja)

Tipičan zadatak nadgledanog učenja je *klasifikacija*. Filtar za neželjenu poštu je dobar primer toga: obučavate ga na većem broju konkretnih primera e-pošte, zajedno sa *klasom* kojoj propada (neželjena ili prihvatljiva poruka), a filtar mora da nauči kako da klasifikuje nove poruke.

Još jedan tipičan zadatak je predviđanje *ciljne* numeričke vrednosti, kao što je cena polovnog automobila, na osnovu zadatog skupa svojstava vozila (kilometraža, starost, marka itd.) koji se zovu *prediktori*. Ta vrsta zadatka zove se *regresija* (eng. *regression*) (slika 1-6).[1] Da biste obučili sistem, treba da mu zadate veći broj primera automobila, zajedno s njihovim prediktorima i njihovim oznakama (odnosno cenama).

U mašinskom učenju *atribut* je tip podataka (na primer „kilometraža"), dok *svojstvo* (eng. *feature*) ima više značenja, u zavisnosti od konteksta, ali uglavnom znači atribut plus njegova vrednost (npr. „kilometraža = 15.000"). Mnogi ljudi mešaju značenje reči *atribut* i *svojstvo*.

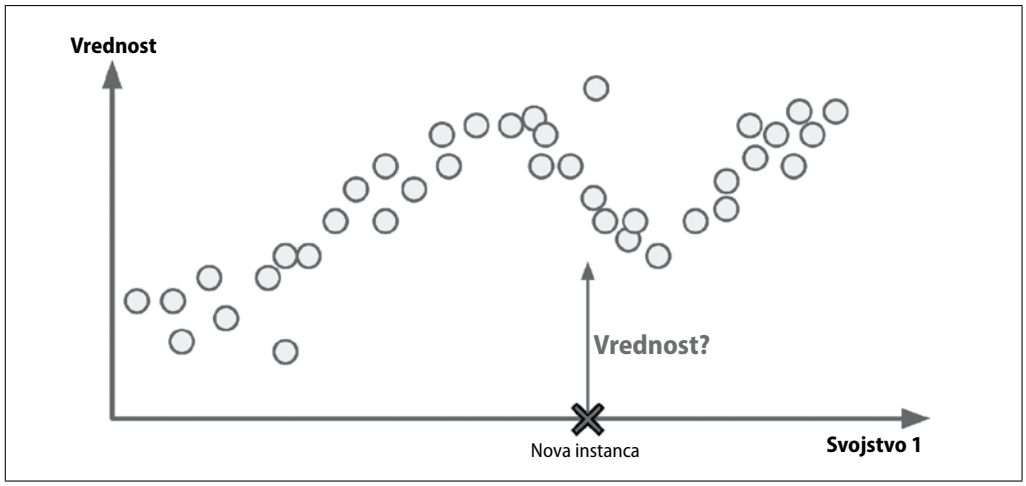
Imajte u vidu da se neki algoritmi za izračunavanje regresije mogu iskoristiti i za klasifikovanje, a važi i obratno. Na primer, *logistička regresija* se često koristi za klasifikovanje, pošto može da pruži vrednost koja odgovara verovatnoći pripadanja datoj klasi (npr. 20% verovatnoće da je u pitanju neželjena poruka).

Nekoliko najvažnijih algoritama za nadgledano učenje (koje ova knjiga opisuje):

- k-najbližih suseda
- Linearna regresija
- Logistička regresija
- Mašine sa vektorom podrške (SVM)

[1] Zanimljiva činjenica: reč *regresija*, koja čudno zvuči, je statistički izraz koji je uveo Francis Galton kada je proučavao činjenicu da su deca visokih osoba sklona tome da budu niža od svojih roditelja. Pošto su deca bila niža, on je to nazvao *regresija ka proseku*. Taj naziv je zatim primenio na metode pomoću kojih je analizirao korelacije između promenljivih.

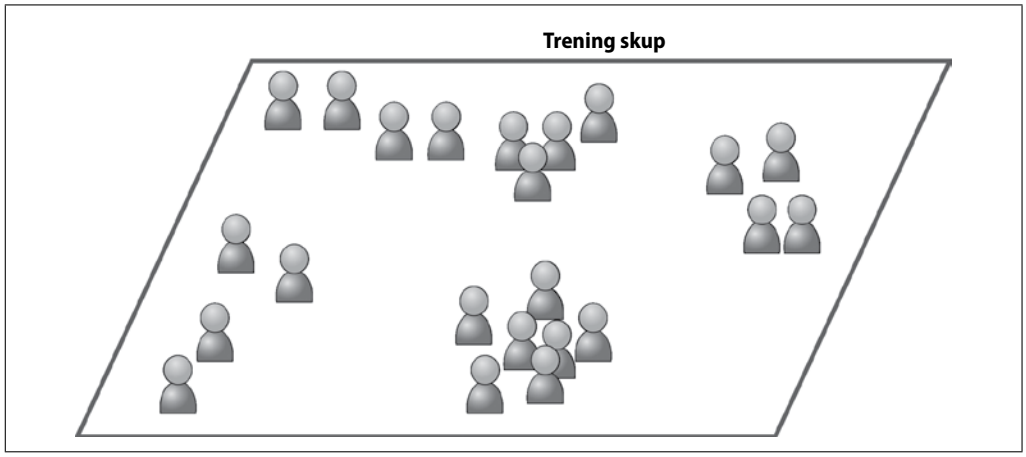
- Stabla odluka i nasumične šume
- Neuronske mreže[2]



Slika 1-6. Problem regresije: predvideti izlaznu vrednost za dato ulazno svojstvo (obično ima više ulaznih svojstava, a ponekad više rezultujućih vrednosti)

Nenadgledano učenje

Kod *nenadgledanog učenja* (eng. *unsupervised learning*), kao što biste i sami pretpostavili, trening skup (skup za obuku) je neoznačen (slika 1-7). Sistem pokušava da uči bez učitelja.



Slika 1-7. Neoznačen trening skup za nenadgledano učenje

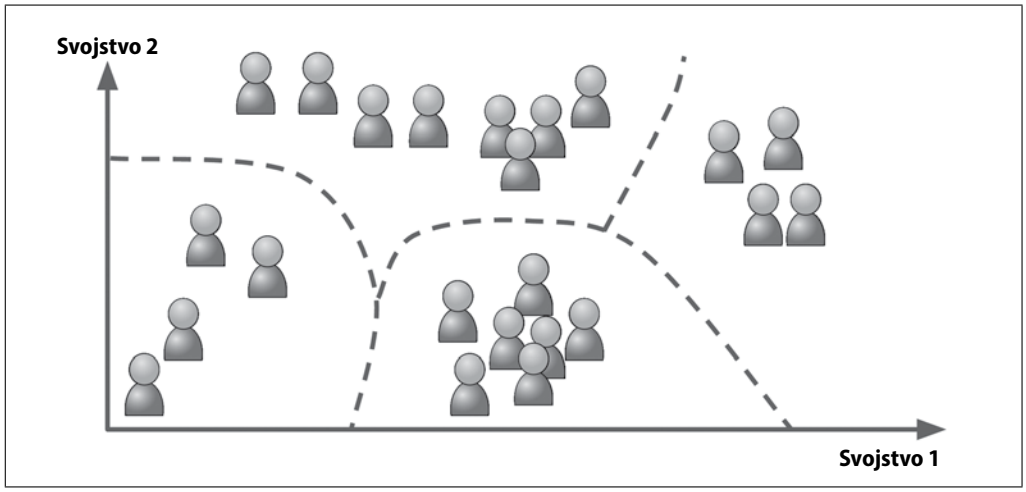
[2] Neke arhitekture neuronskih mreža mogu se svrstati u nenadgledano učenje, kao što su to autoenkoderi i ograničene Bolcmanove mašine. Mogu biti i polunagledane, kao što su to mreže dubokg verovanja i nenadgledano predučenje (eng. *unsupervised pretraining*).

Neki od najvažnijih algoritama za nenadgledano učenje su sledeći (većina je opisana u poglavljima 8 i 9):

- Grupisanje
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Otkrivanje anomalija i otkrivanje novina
  - SVM s jednom klasom
  - Izolaciona šuma
- Vizuelizacija i umanjivanje broja dimenzija
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Učenje na osnovu pravila za asocijacije
  - Apriori
  - Eclat

Recimo da imate veliki broj podataka o posetiocima vašeg bloga. Možda bi trebalo da ih obradite pomoću nekog *algoritma grupisanja* (eng. *clustering algorithm*) da biste otkrili grupe (klastere) sličnih posetilaca (slika 1-8). Ni u jednom slučaju nećete reći algoritmu kojoj grupi pripada dati posetilac: algoritam sam otkriva te veze, bez vaše pomoći. Na primer, može zapaziti da su 40% vaših posetilaca muškarci koji vole stripove i uglavnom čitaju vaš blog tokom popodneva, dok 20% njih su mladi ljubitelji naučne fantastike koji vas posećuju tokom dana vikenda. Ako upotrebite *algoritam hijerarhijskog grupisanja (*eng. *hierarchical clustering algorithm*), on može izdeliti svaku grupu na manje grupe. To vam može pomoći da svoje priloge na blogu naciljate za svaku pojedinačnu grupu.

Algoritmi *vuzuelizovanja* (eng. *visualization algorithms*) takođe su dobri primeri nenadgledanih algoritama učenja: prosleđujete im velike količine podataka koji nemaju oznake, a oni vraćaju 2D ili 3D vizuelizaciju tih podataka koji se lako može iscrtati pomoću plotera (slika 1-9). Ti algoritmi pokušavaju da sačuvaju što je moguće veći deo izvorne strukture podataka (npr. tako što pokušavaju da spreče da zasebne grupe podataka u ulaznom prostoru izgledaju preklopljeno u vizuelizovanom obliku) tako da vi razumete kako su podaci organizovani i možda među njima otkrijete neočekivane šablone.

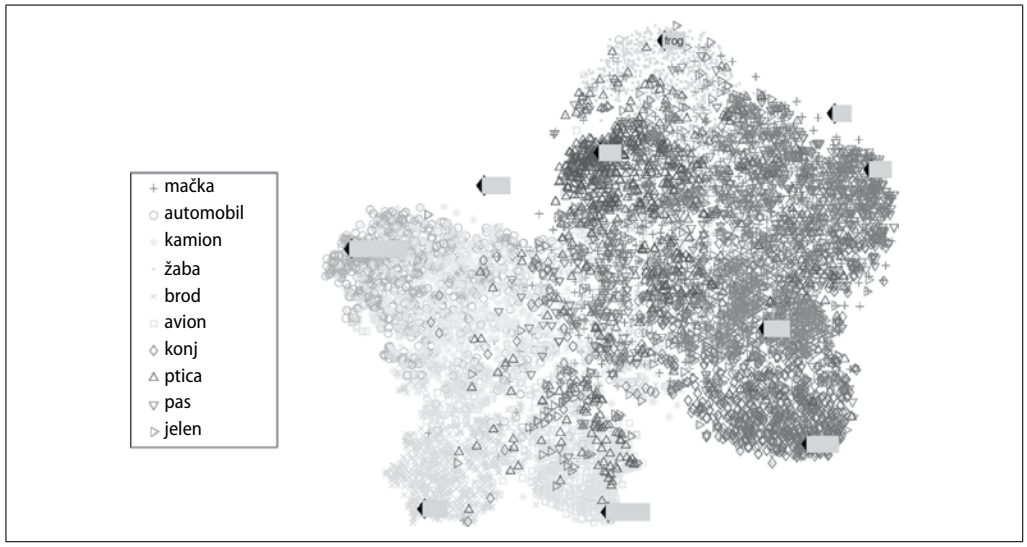


Slika 1-8. Grupisanje

Povezan zadatak je *redukcija dimenzionalnosti* (eng. *dimensionality reduction*), gde je cilj da se podaci pojednostave bez previše gubljenja informacija. Jedan od načina da se to obavi je sjedinjavanje više koreliranih svojstava u jednu. Na primer, kilometraža jednog automobila može veoma zavisiti od njegove starosti, pa će zbog toga algoritam umanjivanja dimenzionalnosti spojiti ta dva svojstva u jedno koje predstavlja istrošenost auta. To se zove *izdvajanje svojstava* (eng. *feature extraction*).

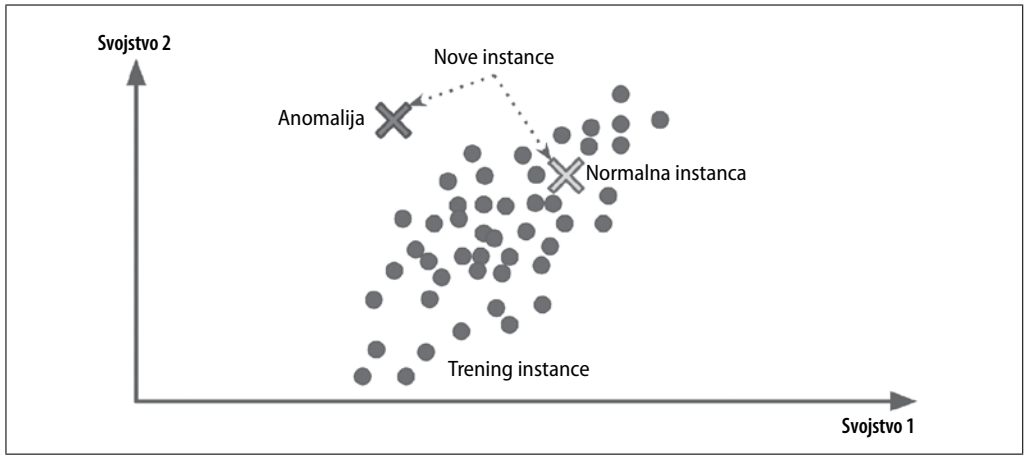
Često je dobra ideja da dimenziju podataka za obuku pokušate da smanjite pomoću nekog algoritma za umanjivanje dimenzionalnosti, pre nego što te podatke prosledite drugom algoritmu mašinskog učenja (kao što je algoritam nadgledanog učenja). On će onda raditi znatno brže, podaci će zauzimati manje mesta na disku i u memoriji, a u nekim slučajevima može čak i raditi bolje.

Još jedan važan nenadgledan zadatak je *otkrivanje anomalija* (eng. *anomaly detection*), na primer, otkrivanje neobičnih transakcija s kreditnim karticama radi sprečavanja prevara, prepoznavanje grešaka pri proizvodnji ili automatsko uklanjanje besmislenih podataka (anomalija) iz datog skupa pre njihove obrade pomoću drugog algoritma učenja. Pošto tokom učenja sistem dobija uglavnom normalne instance, on nauči da ih prepoznaje; zatim, kada vidi nov primer, može da razlikuje da li on izgleda kao normalan ili je verovatno u pitanju neka anomalija (slika 1-10). Vrlo sličan zadatak je *otkrivanje novine* (eng. *novelty detection*): cilj je da se otkriju novi primeri koji izgledaju drugačije od svih primera u trening skupu. To zahteva veoma „čist" trening skup, bez jednog jedinog primera koju biste želeli da algoritam kasnije otkriva. Na primer, ako u trening skupu imate hiljade slika pasa, a samo 1% tih slika predstavljaju čivave, algoritam otkrivanja novina ne bi trebalo da obrađuje nove slike čivava kao novine. S druge strane, algoritam otkrivanja anomalija može smatrati da su ti psi toliko retki i toliko drugačiji od drugih pasa da će ih verovatno klasifikovati kao anomalije (bez uvrede čivavama).



Slika 1-9. Primer t-SNE vizuelizacije koja ističe semantičke klastere[3]
(slika u boji na www.mikroknjiga.rs)

I najzad, još jedan čest nenadgledan zadatak je *učenje na osnovu pravila za asocijacije* (eng. *association rule learning*), gde je cilj obrada velike količine podataka radi otkrivanja zanimljivih veza između pojedinih atributa. Na primer, vi ste vlasnik supermarketa. Primena odgovarajućeg pravila za asocijacije na vaše podatke može otkriti da ljudi koji kupe sos za roštilj i čips od krompira imaju tendenciju da kupe i šniclu. Iz tih razloga, možda bi trebalo da te artikle postavite na policama jedan u blizini drugog.

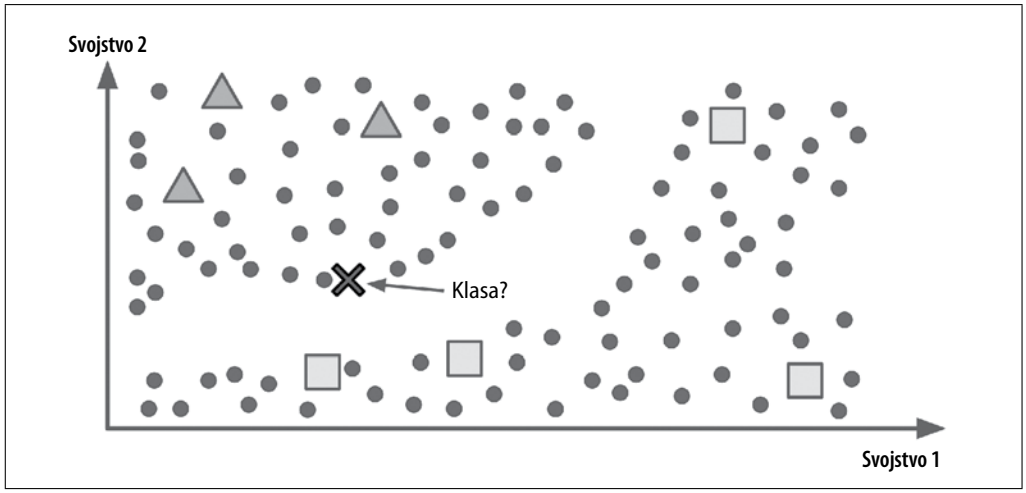


Slika 1-10. Otkrivanje anomalija

[3] Obratite pažnju koliko su životinje jasno razdvojene od vozila i kako su konji blizu jelena, ali daleko od ptica. Ova slika je reprodukovana uz dozvolu autora Richard Socher et al, „Zero-Shot Learning Through Cross-Modal Transfer"; *Proceedings of the 26th International Conference on Neural Information Processing Systems* 1 (2013): 935–943.

Polunadgledano učenje

Pošto označavanje podataka obično zahteva vreme i ima svoju cenu, često ćete imati mnogo primera (instanci) bez oznake i nešto primera sa oznakama. Neki algoritmi mogu da rade s podacima koji su samo delimično označeni. To se zove *polunadgledano učenje* (eng. *semisupervised learning*) (slika 1-11).



Slika 1-11. Polunadgledano učenje sa dve klase (trouglovi i kvadrati): primeri bez oznaka (kružići) pomažu da se novi primer (krstić) razvrsta u klasu trougao umesto u klasu kvadrat, uprkos tome što se krstić nalazi bliže kvadratima sa oznakama

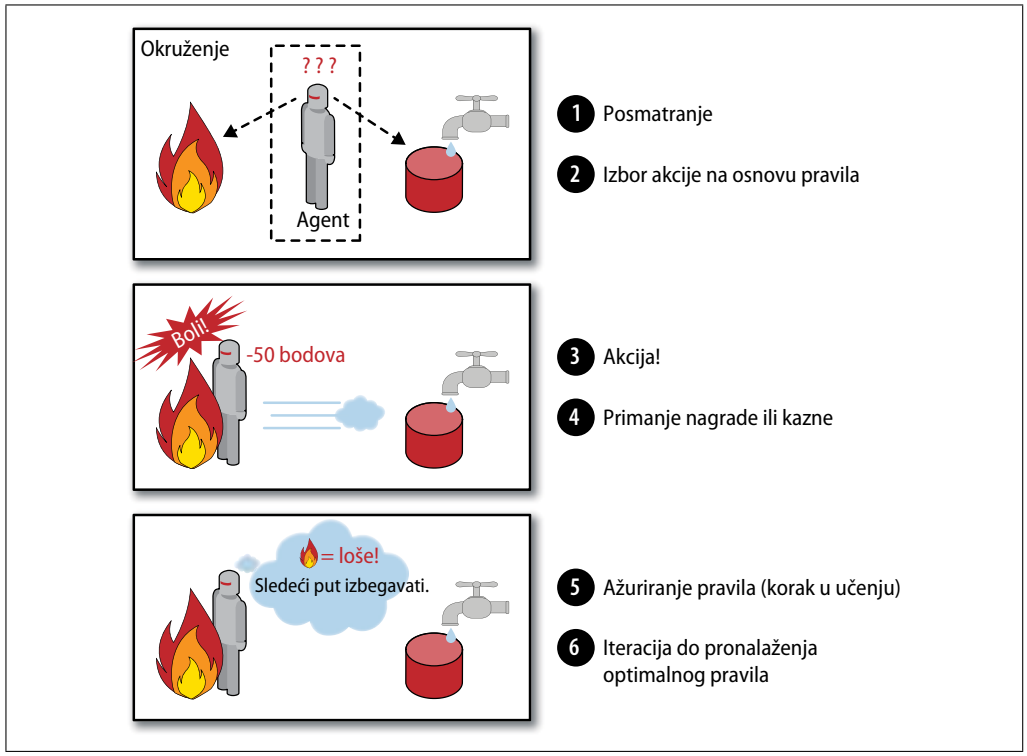
Dobar primer toga su neki od servisa za čuvanje fotografija, kao što je Google Photos. Pošto na takav servis prebacite sve svoje porodične fotografije, on automatski prepoznaje da se ista osoba A nalazi na fotografijama 1, 5 i 11, a druga osoba B se prikazuje na slikama 2, 5 i 7. To je nenadgledani deo algoritma (grupisanje). Nakon toga sve što sistemu treba je da mu vi kažete ko su ti ljudi. Samo pridružite po jednu oznaku svakoj osobi[4] pa će sistem biti u stanju da prepozna svakoga na fotografiji, što je korisno kod pretraživanja fotografija.

Većina algoritama za polunadgledano učenje su kombinacije nenadgledanih i nadgledanih algoritama. Na primer, *mreže dubokog verovanja* (eng. *deep belief networks, DBNs*) zasnivaju se na nenadgledanim komponentama koje se zovu *ograničene Bolcmanove mašine* (eng. *restricted Boltzmann machines, RBMs*), nadovezane jedna na drugu. RBM komponente uče sekvencijalno na nenadgledan način, a zatim se ceo sistem preciznije podešava pomoću nadgledanih tehnika učenja.

[4] Tako je kada sistem radi savršeno. U praksi često formira nekoliko grupa po osobi, a ponekad i brka dve osobe koje izgledaju slično, pa zato može biti potrebno da zadate više oznaka za istu osobu i da neke grupe ručno očistite.

Forsirano učenje

Forsirano učenje ili *učenje uslovljavanjem* (eng. *reinforcement learning*) je sasvim drugačija „zver". Sistem koji uči, koji se u ovom kontekstu zove *agent*, može da posmatra svoje okruženje, bira i izvršava akcije, za koje dobija *nagrade* (eng. *rewards*) ili *kazne* (eng. *penalties*) u obliku negativnih nagrada, kao što prikazuje slika 1-12. Zatim sistem mora sam da nauči koja je najbolja strategija, koja se zove *pravilo* (eng. *policy*), da tokom vremena dobije najviše nagrada. Pravilo definiše koju akciju agent treba da izabere kada se nađe u datoj situaciji.



Slika 1-12. Forsirano učenje

Na primer, mnogi roboti koriste algoritme za forsirano učenje da bi naučili da hodaju. Program AlphaGo kompanije DeepMind takođe je dobar primer forsiranog učenja. Bio je tema naslova novina u maju 2017, kada je pobedio Ke Jiea, svetskog šampiona u igri Go. Do pobedničke strategije je došao analiziranjem miliona partija, a zatim i igranjem velikog broja partija protiv samog sebe. Imajte u vidu da je učenje bilo isključeno tokom igranja protiv šampiona; AlphaGo je samo primenjivao pravilo koje je naučio.

Paketno i onlajn učenje

Još jedna osnova za razvrstavanje sistema mašinskog učenja je to da li je sistem u stanju ili nije da uči postupno iz toka dolazećih podataka.

Paketno učenje

Kod *paketnog učenja* (eng. *batch learning*), sistem nije u stanju da uči postepeno: mora se obučavati na osnovu svih raspoloživih podataka. Pošto to uglavnom zahteva mnogo vremena i računarskih resursa, obično se radi oflajn. Prvo se sistem obučava, a zatim stavlja u redovnu upotrebu i radi tako da više ne uči, nego samo primenjuje ono što je ranije naučio. To se zove *oflajn* tj. *učenje van mreže* (eng. *offline learning*).

Ako želite da sistem s paketnim učenjem upozna nove podatke (kao što je nova vrsta neželjenih poruka), potrebno je da novu verziju sistema obučite ispočetka s kompletnim podacima (ne samo s novim, nego i sa starim), a zatim morate isključiti staru verziju sistema i zameniti je novom.

Srećom, ceo postupak obučavanja, ispitivanja i stavljanja u redovnu upotebu jednog sistema mašinskog učenja može se prilično lako automatizovati (kao što prikazuje slika 1-3), tako da se i sistem s paketnim učenjem može prilagoditi promeni. Samo ažurirajte podatke i obučite novu verziju sistema ispočetka koliko god puta to bude potrebno.

To rešenje je jednostavno i često sasvim dobro, ali pošto obuka sistema na kompletnom skupu podataka može zahtevati više sati, novu verziju sistema biste uglavnom obučavali svakih 24 sata ili čak samo jednom nedeljno. Ako vaš sistem treba da se prilagođava podacima koji se brzo menjaju (npr. za predviđanje cena na berzi), potrebno vam je reaktivnije rešenje, s bržim odzivom.

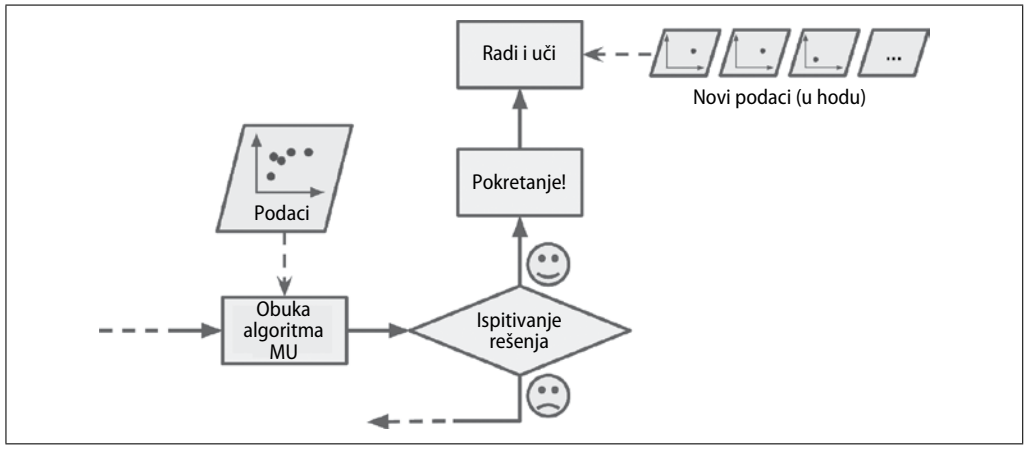
Osim toga, obuka na kompletnom skupu podataka zahteva mnogo računarskih resursa (CPU, memorija, prostor na disku, U/I operacije s diskom, U/I operacije s mrežom itd.). Ako imate veliku količinu podataka i svoj sistem automatizujete tako da se on svakog dana ponovo obučava od početka, to će vas vremenom koštati mnogo novca. Ako je količina podataka ogromna, upotreba nekog algoritma za paketno učenje može biti čak i nemoguća.

I najzad, ako vaš sistem treba da bude u stanju da samostalno uči i raspolaže ograničenim resursima (npr. aplikacija za pametan telefon ili robot koji istražuje Mars), onda manipulisanje ogromnim količinama trening podataka i trošenje velikih količina resursa za višesatnu obuku sistema svakog dana može baš pokvariti zabavu.

Srećom, bolje rešenje u svakom od takvih slučajeva je upotreba algoritma koji su u stanju da uče postepeno.

Onlajn učenje

Kod *onlajn učenja* (eng. *online learning*), sistem obučavate postepeno tako što mu sekvencijalno zadajete primere podataka, pojedinačno ili u malim grupama koje se zovu *mini paketi* (eng. *mini-batches*). Pošto je svaki korak učenja kratak i jeftin, sistem može da uči na novim podacima u hodu, kako oni pristižu (slika 1-13).



Slika 1-13. Kod onlajn učenja, model se prvi put obučava i stavlja u proizvodnju, a zatim nastavlja da uči kako stižu novi podaci

Onlajn učenje je odlično za sisteme koji primaju podatke u obliku neprekidnog toka (npr. cene na berzi) i treba da se prilagođavaju promenama brzo ili samostalno. To je takođe dobra opcija ako imate ograničene računarske resurse: pošto sistem za onlajn učenje obradi sve nove primere podataka, oni mu više ne trebaju, pa ih zato možete odbaciti (osim ako želite mogućnost da sistem vratite u neko ranije stanje i ponovo mu „pustite" podatke). Tako se može uštetedeti veliki prostor na disku.

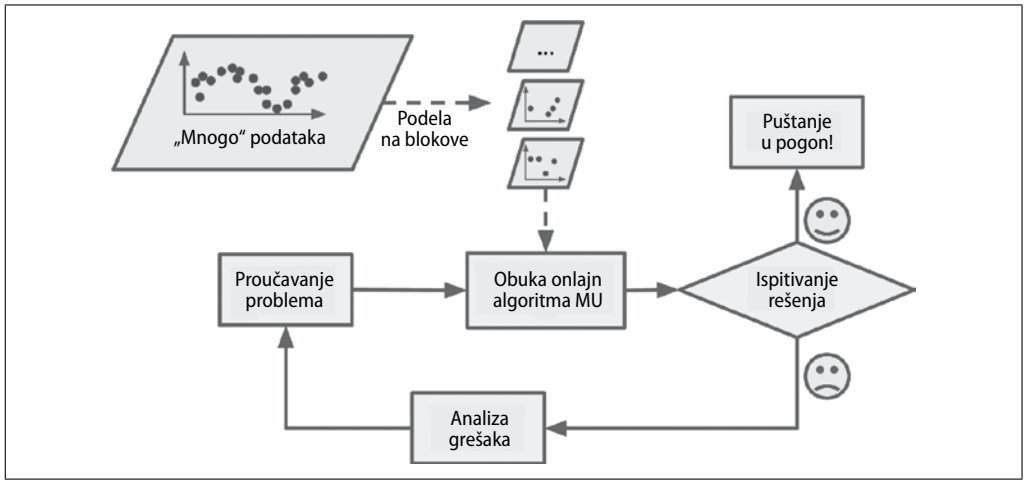
Algoritmi za onlajn učenje mogu se iskoristiti i za obuku sistema sa veoma obimnim skupovima podataka koji ne mogu da stanu u glavnu memoriju sistema. To se zove *učenje izvan jezgra* (eng. *out-of-core learning*). Algoritam učitava deo podataka, obavlja korak učenja s tim podacima, a zatim ponavlja ceo postupak dok tako ne obradi sve podatke (slika 1-14).

Učenje izvan jezgra se obično obavlja oflajn (tj. ne na sistemu u redovnoj upotrebi), pa zato izraz *onlajn učenje* može da zbuni. Zamislite ga kao *postepeno učenje*.

Važan parametar sistema za onlajn učenje je koliko brzo treba da se prilagodi izmenjenim podacima: to se zove *brzina učenja* (eng. *learning rate*). Ako zadate visoku brzinu učenja, vaš sistem će se brzo prilagođavati novim podacima, ali će imati i sklonost da brzo zaboravlja stare podatke (ne biste želeli da filtar za neželjene poruke izdvaja samo najnovije vrste takvih poruka koje mu pokažete). Nasuprot tome, ako zadate nisku brzinu učenja, sistem će imati veću inerciju, odnosno učiće sporije, ali će biti i manje osetljiv na šum u novim podacima ili na sekvence nereprezentativnih podataka (anomalija).

Veliki izazov kod onlajn učenja je to da kada sistem dobije loše podatke, njegove performanse se postepeno pogoršavaju. Ako je sistem u redovnoj upotrebi, vaši klijenti će to primetiti. Na primer, pogrešni podaci mogu poticati od neispravnog senzora na robotu, ili od nekoga ko

namerno šalje pogrešne podatke mašini za pretraživanje pokušavajući da postigne visok nivo među rezultatima pretraživanja. Da biste umanjili taj rizik, potrebno je da redovno pratite svoj sistem i brzo isključite učenje (i eventualno vratite sistem u neko ranije radno stanje) ako otkrijete pogoršanje njegovih performansi. Možda bi trebalo i da nadgledate ulazne podatke i reagujete na nenormalne podatke (npr. pomoću nekog algoritma za otkrivanje anomalija).



Slika 1-14. Primena onlajn učenja za obradu ogromnih skupova podataka

Učenje na primerima nasuprot učenju na modelu

Još jedan oblik kategorizovanja sistema mašinskog učenja je način na koji oni *generalizuju*. Većina poslova koji se obavljaju pomoću mašinskog učenja svodi se na predviđanja. To znači da pošto mu zadate više primera za obuku, trebalo bi da sistem bude u stanju da daje dobra predviđanja (generalizuje) za primere koje ranije nije video. Postizanje dobrih performansi sa podacima za obuku jeste dobro, ali nije dovoljno; pravi cilj je dobra obrada novih primera.

Postoje dva glavna pristupa problemu generalizovanja: učenje na konkretnim primerima i učenje na modelu.

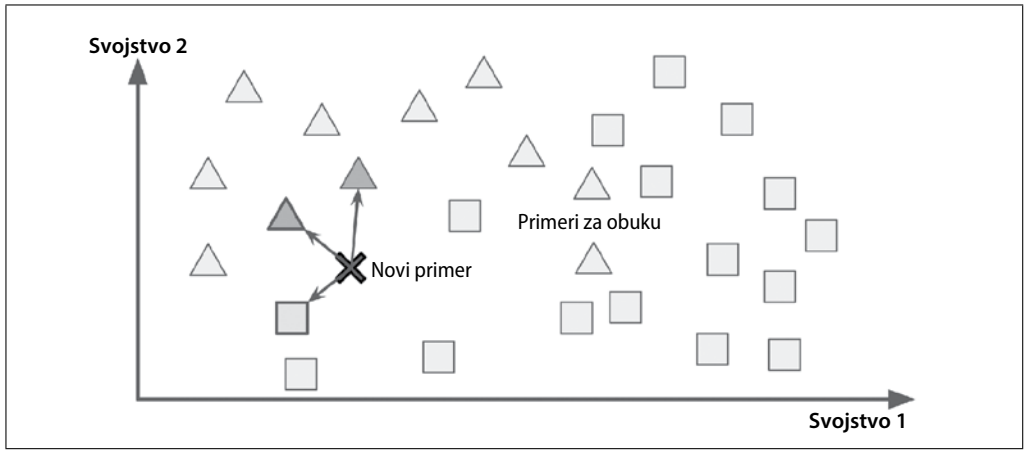
Učenje na primerima

Možda najtrivijalniji oblik učenja je učenje napamet. Ako bi trebalo da napravite filtar za neželjene poruke koji tako radi, on bi samo izdvajao poruke koje su identične onima koje su korisnici ranije označili kao neželjene, što nije najgore rešenje, ali svakako nije ni najbolje.

Umesto da samo izdavaja e-poruke koje su identične poznatim neželjenim porukama, vaš filtar biste mogli programirati tako da izdvaja i poruke koje su im vrlo slične. Za to je potrebno definisati *meru sličnosti* (eng. *measure of similarity*) između dve e-poruke. Jedna (veoma jednostavna) mera sličnosti između dve e-poruke mogao bi da bude broj reči koje su

im zajedničke. Sistem bi onda obeležavao kao neželjenu poruku svaku u kojoj se pojavljuje određen broj istih reči kao u poznatoj neželjenoj poruci.

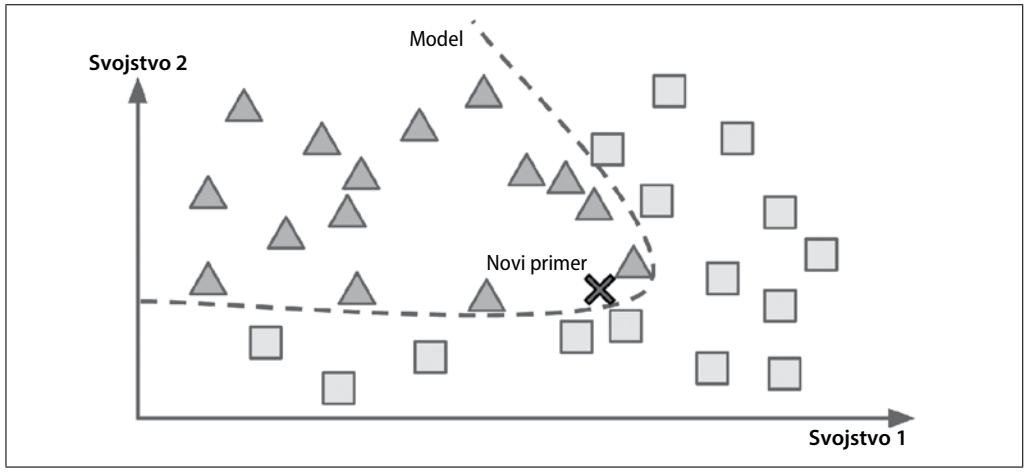
Ovo se zove *učenje na primerima* (eng. *instance-based learning*): sistem uči zadate primere napamet, a zatim ih generalizuje na nove slučajeve pomoću date mere sličnosti tako što ih poredi s naučenim primerima (ili nekim njihovim podskupom). Na primer, na slici 1-15 novi primer bi bio klasifikovan kao trougao zato što većina njemu sličnih primera pripada toj klasi.



Slika 1-15. Učenje na primerima

Učenje na modelu

Drugi način da se generalizuje skup početnih primera je da napravite model tih primera i da zatim na osnovu tog modela pravite *predviđanja* (eng. *predictions*). To se zove *učenje na modelu* (eng. *model-based learning*) (slika 1-16).



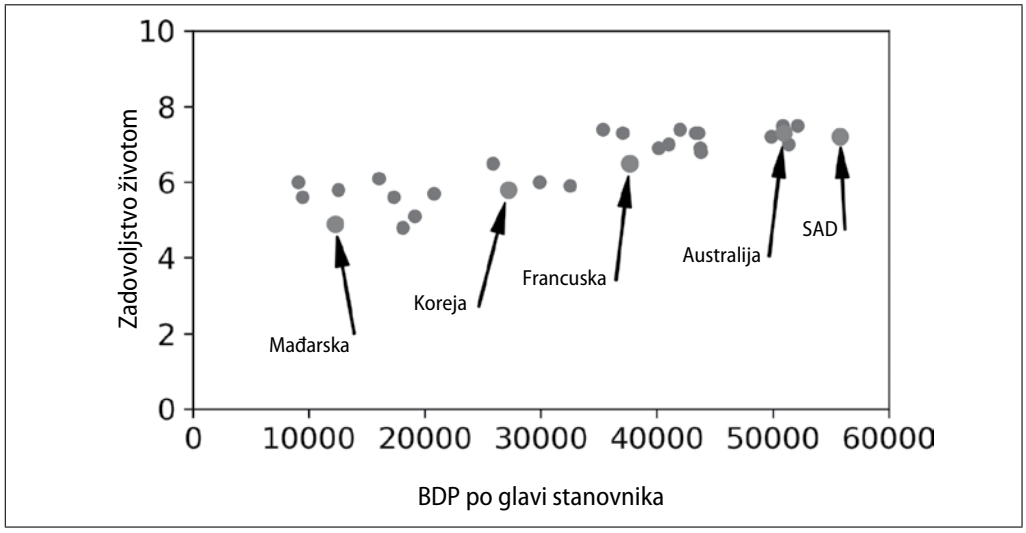
Slika 1-16. Učenje na modelu

Na primer, želite da znate da li novac čini ljude srećnim, pa zato sa veb lokacije organizacije OECD (*https://homl.info/4*) preuzimate podatke Better Life Index (indeks boljeg života), kao i statistike o bruto domaćem proizvodu (BDP) po glavi stanovnika, sa veb lokacije MMF-a (*https://homl.info/5*). Zatim spojite te tabele i sortitrate po BDP-u po glavi stanovnik. Tabela 1-1 prikazuje deo podataka koje dobijate.

Tabela 1-1. Da li novac čini ljude srećnijim?

Država	BDP po glavi stanovnika (SAD)	Zadovoljstvo životom
Mađarska	12,240	4,9
Koreja	27,195	5,8
Francuska	37,675	6,5
Australija	50,962	7,3
SAD	55,805	7,2

Nacrtajmo sada dijagram podataka za ove države (slika 1-17).



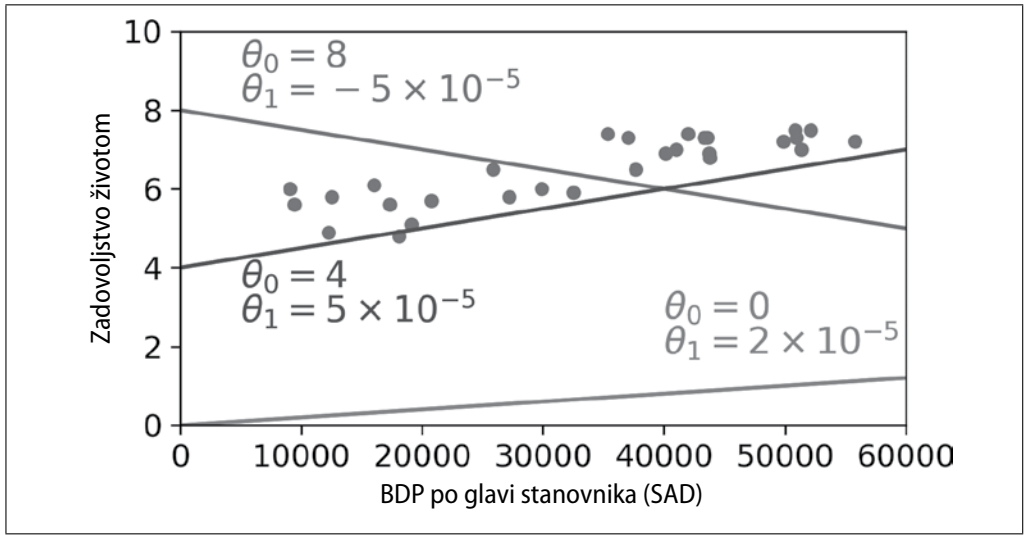
Slika 1-17. Da li ovde uočavate nekakav trend?

Ovde je jasno vidljiv trend! Uprkos tome što su podaci *zagađeni šumom* (tj. delimično nasumični), izgleda kao da zadovoljstvo životom raste manje ili više linearno srazmerno rastu BDP-a po glavi stanovnika u državi. Zato se opredeljujete da zadovoljstvo života modelujete u obliku linearne funkcije BDP-a po glavi stanovnika. Taj korak se zove *biranje modela*: izabrali ste *linearni model* zadovoljstva života, koji ima samo jedan atribut, BDP po glavi stanovnika (jednačina 1-1)

Jednačina 1-1 Jednostavan linearni model

zadovoljstvo_životom = $\theta_0 + \theta_1 \times$ BDP_po_ glavi stanovnika

Ovaj model ima dva *parametra modela* (eng. *model parameters*), θ_0 i θ_1.[5] Menjanjem vrednosti tih parametara možete podesiti da vaš model predstavlja proizvoljnu linearnu funkciju, kao što je prikazano na slici 1-18.



Slika 1-18. Nekoliko mogućih linearnih modela

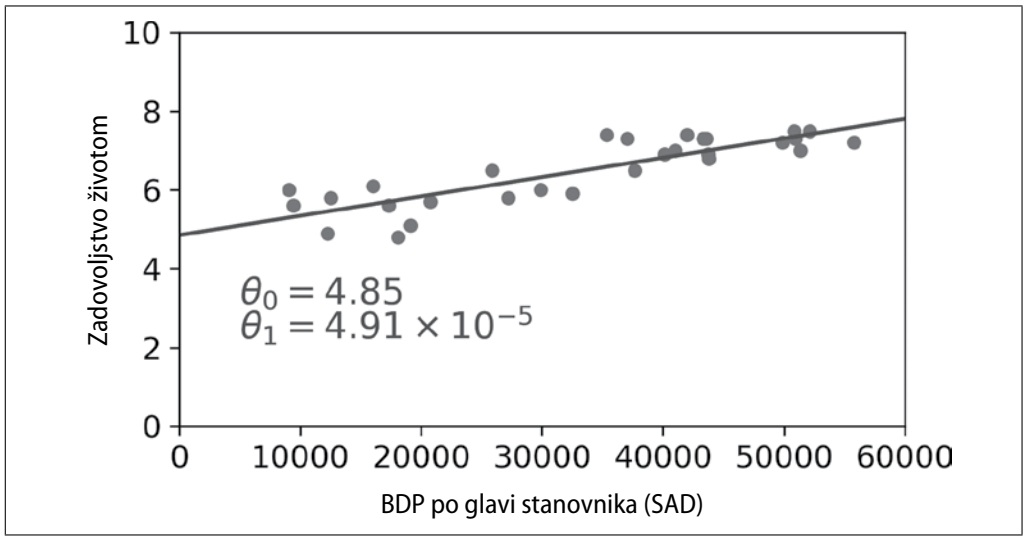
Pre upotrebe modela, potrebno je da zadate vrednosti parametrima θ_0 i θ_1. Kako možete znati koje će vrednosti učiniti da vaš model najbolje radi? Da biste odgovorili na to pitanje, treba da zadate meru performansi. Za tu namenu možete definisati *funkciju korisnosti* (eng. *utility function*) ili *funkciju uklapanja* (eng. *fitness function*), koja meri koliko je vaš model *dobar*, ili možete definisati *funkciju gubitka* (eng. *cost function*), koja meri koliko je model *loš*. Za probleme linearne regresije, ljudi obično koriste određenu funkciju gubitka koja meri razliku između predviđanja linearnog modela i primera za obuku. Cilj je minimizovanje te razlike.

Tu uskače algoritam linearne regresije, kojem zadajete vaše primere za obuku, a on pronalazi vrednosti parametara koje najbolje uklapaju linearni model u vaše podatke. To se zove *obuka* (eng. *training*) modela. U našem slučaju, algoritam pronalazi da su optimalne vrednosti parametara $\theta_0 = 4{,}85$ i $\theta_1 = 4{,}91 \times 10^{-5}$.

[5] Važi konvencija prema kojoj se parametri često predstavljaju kao grčko slovo θ (teta).

Zbunjujuće je to što se ista reč „model" može odnositi na *vrstu modela* (eng. *type of model*), na primer, linearna regresija), na *potpuno definisan model arhitekture* (eng. *fully specified model architecture*), na primer, linearna regresija s jednim ulaznim i jednim izlaznim skupom, ili na *konačni obučeni model* (eng. *final trained model*), spreman za upotrebu, koji izračunava predviđanja (npr. linearna regresija s jednim ulaznim i jednim izlaznim skupom, gde je $\theta_0 = 4{,}85$ i $\theta_1 = 4{,}91 \times 10^{-5}$). Postupak biranja modela se sastoji od biranja vrste modela i zadavanja celokupne arhitekture. Obuka modela znači izvršavanje određenog algoritma radi utvrđivanja vrednosti parametra modela za koje se model najbolje uklapa sa podacima za obuku (i za koje očekujemo da ćemo dobijati dobra predviđanja s novim podacima).

Sada se model uklapa najbolje moguće (za linearni model) sa podacima za obuku, što je vidljivo na slici 1-19.



Slika 1-19. Linearni model koji se najbolje uklapa sa podacima za obuku

Sada ste konačno spremni da upotrebite model za izračunavanje predviđanja. Na primer, želite da znate koliko su srećni građani Kipra, a OECD podaci ne sadrže taj podatak. Srećom, možete iskoristiti svoj model da biste izračunali dobro predviđanje: potražićete BDP po glavi stanovnika na Kipru, pronaći da je to $22.587, a zatim ćete primeniti model i otkriti da je zadovoljstvo životom tamo otprilike $4{,}85 + 22.587 \times 4{,}91 \times 10^{-5} = 5{,}96$.

Da bi vam otvorio apetit, primer 1-1 prikazuje Python kôd koji učitava podatke, priprema ih,[6] ispisuje tačkasti dijagram kao oblik vizuelizovanja, a zatim obučava linearni model i izračunava predviđanje.[7]

Primer 1-1 Obuka i izvršavanje linearnog modela pomoću Python biblioteke Scikit-Learn

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model

# Učitavanje podataka
oecd_bli = pd.read_csv("oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv("gdp_per_capita.csv", thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")

# Priprema podataka
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Vizuelizovanje podataka
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Izbor linearnog modela
model = sklearn.linear_model.LinearRegression()

# Obučavanje modela
model.fit(X, y)

# Izračunavanje predviđanja za Kipar
X_new = [[22587]]  # Kiparski BDP po
print(model.predict(X_new)) # rezultat je [[ 5.96242338]]
```

[6] Definicija funkcije `prepare_country_stats()` ovde nije prikazana (videti Jupyter Notebook pridružen izvornom izdanju ove knjige ako vas zanimaju detalji). To je samo običan kôd iz biblioteke pandas koji povezuje podatke o zadovoljstvu životom sa OECD-ove veb lokacije, s podacima o BDP-u po glavi stanovnika na MMF-ovoj veb lokaciji.

[7] Sasvim je u redu ako još ne razumete ceo kôd; u poglavljima koja slede predstavićemo biblioteku Scikit-Learn.

Da ste umesto linearnog, upotrebili neki algoritam učenja na primerima, otkrili biste da najbliži BDP po glavi stanovnika kiparskom ima Slovenija ($20.732), a pošto OECD podaci pokazuju da je zadovoljstvo životom u Slovenuiji 5,7, predvideli biste da je i na Kipru zadovoljstvo života 5,7. Ako podrobnije razmotrite te podatke i pogledate dve tome najbliže države, naći ćete Portugal i Španiju, gde je zadovoljstvo životom 5,1 odonosno 6,5. Ako izračunate prosek te tri vrednosti, dobićete 5,77, što je prilično blizu vašem predviđanju na osnovu modela. Ovaj jednostavan algoritam se zove regresija *k-najbližih suseda* (eng. *k-nearest neighbors*) (u ovom primeru, $k = 3$).

Zamena modela linearne regresije s regresijom k-najbližih suseda u prethodnom kodu svodi se na zamenu naredna dva reda:

```
import sklearn.linear_model
model = sklearn.linear_model.LinearRegression()
```

sledećim:

```
import sklearn.neighbors
model = sklearn.neighbors.KNeighborsRegressor(n_neighbors=3)
```

Ako je sve prošlo kako treba, vaš model će izračunavati dobra predviđanja. Ako nije, može biti potrebno da zadate više atributa (stopa zaposlenosti, kvalitet zdravstvenih usluga, stopa zagađenosti vazduha itd.) da biste dobili veću količinu ili kvalitetnije podatke za obuku, ili ćete možda morati da izaberete moćniji model (npr. model polinomijalne regresije).

Da rezimiramo:

- Proučili ste podatke.
- Izabrali ste model.
- Obučili ste ga na podacima za obuku (npr. tako što je algoritam učenja izračunao vrednosti parametara modela koje minimizuju određenu funkciju gubitka).
- I najzad, primenili ste model za izračunavanje predviđanja za nove slučajeve, to se zove *izvođenje* (eng. *inference*), nadajući se da će model dobro generalizovati.

Ovako izgleda tipičan projekat mašinskog učenja. U poglavlju 2 steći ćete praktično iskustvo na takvom projektu koji ćete pratiti od početka do kraja.

Dosad smo objasnili prilično mnogo toga: sada znate šta je suština mašinskog učenja, zbog čega je ono korisno, koje su neke među najuobičajnijim kategorijama sistema za MU i kako se odvija tipičan projekat. Sada ćemo pogledati šta može krenuti naopako tokom učenja i sprečiti vas da izračunavate tačna predviđanja.

Glavni izazovi mašinskog učenja

Ukratko, pošto je vaš glavni zadatak da izaberete odgovarajući algoritam učenja i obučite ga na nekim podacima, dve stvari koje mogu krenuti naopako jesu „loš algoritam" i „loši podaci". Krenućemo od primera loših podataka.