SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5941

Usporedba metoda strojnog učenja s jasnim i nejasnim tumačenjem modela

Ardian Pantina

Zagreb, lipanj 2019.

Posebna zahvala mentoru Alanu Joviću, na korisnim savjetima i pomoći prilikom izrade ovog završnog rada.

SADRŽAJ

1. Uvod **1**

2. Algoritmi strojnog učenja **3**

2.1. Stabla odluke 3

2.2. Algoritam K-najbližih susjeda 6

2.3. Ansambli stabala odluke 8

2.4. Strojevi s potpornim vektorima 10

2.5. Neuronske mreže 12

3. Interpretabilnost i točnost metoda strojnog učenja **14**

3.1. Učenje i testiranje metoda 14

3.2. Unakrsna validacija 15

3.3. Interpretabilnost metoda strojnog učenja 16

4. Vrednovanje algoritama strojnog učenja koristeći Scikit-Learn **18**

4.1. Knjižnica scikit-learn 18

4.2. Aplikacija za učitavanje podataka, učenje i testiranje 19

4.3. Eksperimentalni rezultati vrednovanja algoritama 21

5. Zaključak **24**

Literatura **25**

1. Uvod

Strojno učenje je područje računalne znanosti koje s vremenom nudi sve zanimljivija i učinkovitija rješenja za razne probleme u stvarnom životu. Od autonomnih vozila i prilagodljivih filtra poruka e-pošte, pa sve do sustava za prijedlog medijskih sadržaja, algoritmi strojnog učenja imaju sve važniju ulogu u vođenju procesa ljudskog razumijevanja i donošenja odluka. Mnogi problemi koji su donedavno bili presloženi da bi se algoritamski riješili, poput raspoznavanja govora, lica ili slika, pokazali su se mogućim za generalizaciju koristeći metode poput neuronskih mreža. Ogromni skupovi podataka koji su se činili nepovezanim ili previše složenim da bi se od njih izvukao nekakav zaključak, danas se mogu dubinski analizirati kako bi se iz njih otkrilo novo znanje. Područja poput statističke analize podataka i obrade prirodnog jezika doživjela su strahovit rast upravo zbog otkrića novih algoritama i modela pomoću kojih računala mogu sama donositi zaključke i nove informacije na temelju postojećih podataka.

Rad će se fokusirati na nadzirane sustave strojnog učenja, odnosno one sustave koji se uče na način da ulazni podaci sustava sadržavaju ciljni atribut. Zatim se na temelju tih podataka može predvidjeti vrijednost ciljnog atributa za potpuno nove podatke, kod kojih je vrijednost tog ciljnog atributa nepoznata. Formalno, podaci su u obliku $(ulaz, izlaz) = (\mathbf{x}, y)$, i potrebno je naći preslikavanje $\hat{y} = f(\mathbf{x})$. Ako je y kontinuirana vrijednost, promatramo probleme regresije, a ako je y diskretna vrijednost, radi se o problemima klasifikacije, koji se pobliže razmatraju u sklopu ovog rada. Dobar primjer klasifikacije jest tzv. *spam filter*, koji se uči na većem skupu malicioznih i bezopasnih poruka, te zatim može nove poruke svrstati u odgovarajuću kategoriju. U tom slučaju y može predstavljati diskretnu binarnu vrijednost koja pokazuje da li je e-pošta zlonamjernog sadržaja ili ne.

Problem koji će se promatrati jest činjenica da većina metoda strojnog učenja ima relativno nejasno tumačenje, budući da u pozadini koriste kompleksne matematičke izračune i algoritme. Međutim, upravo zbog te kompleksnosti ti algoritmi generalno postižu bolje rezultate nad skupovima podataka. Dokaz tome su natjecanja u strojnom učenju poput *Kagglea*, gdje sudionici imaju tendenciju koristiti ansamble više različitih modela, što uz veću preciznost dodatno otežava razumijevanje. Razmotrit ćemo zašto je bitno razumjeti kako funkcioniraju ove metode i sustavi, budući da imaju sve veću ulogu u našoj svakodnevnici.

U praktičnom dijelu ovog rada će se demonstrirati aplikacija koja koristi knjižnicu *scikit-learn* i njene algoritme strojnog učenja nad stvarnim skupovima podataka za ocjenu točnosti i vizualizaciju rezultata. Uz to, napravit će se usporedba više algoritama, jasnih i nejasnih, nad istim skupovima podataka kako bi se usporedili njihovi rezultati.

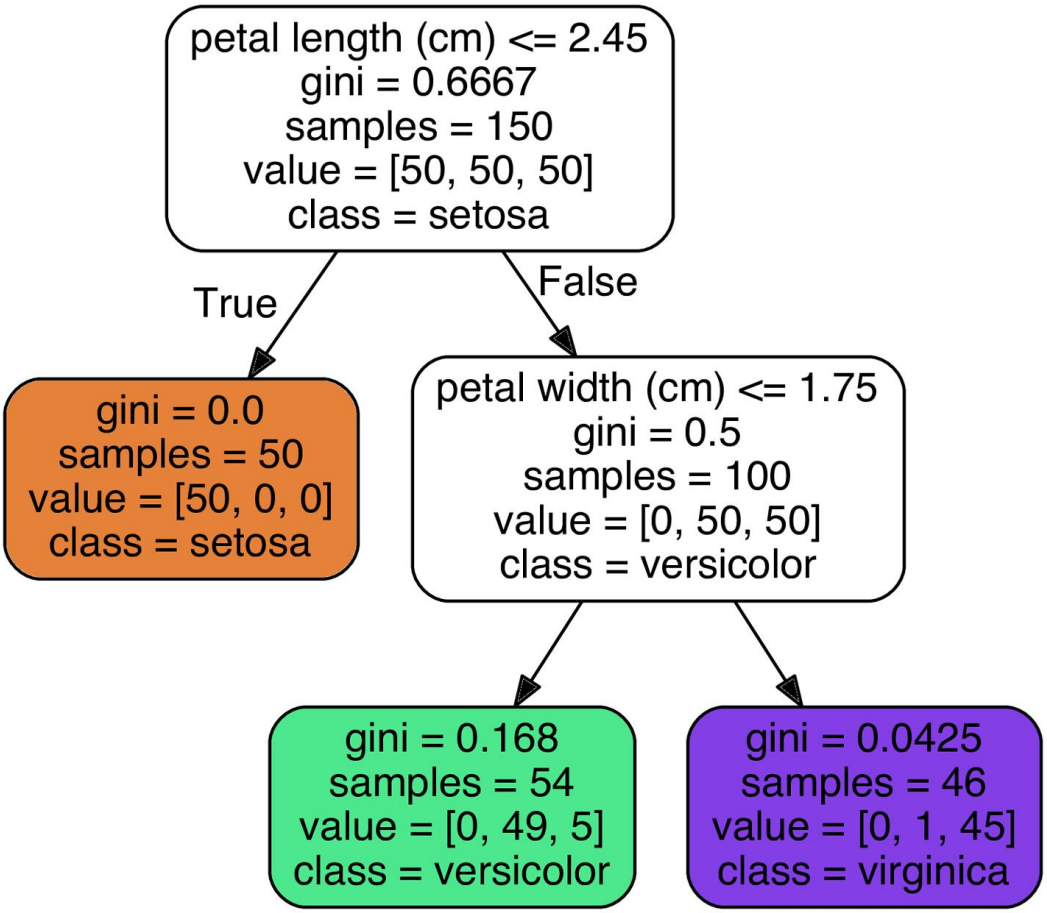
2. Algoritmi strojnog učenja

U ovom poglavlju opisat će se odabrani algoritmi strojnog učenja čija će se funkcionalnost kasnije demonstrirati u praktičnom dijelu. Ti algoritmi redom su stabla odluke i algoritam k-najbližih susjeda, za koje možemo reći da imaju jasno značenje modela, te s druge strane ansamble stabala odluke, strojeve s potpornim vektorima i neuronske mreže, za koje se smatra da imaju značajno veću težinu interpretacije.

2.1. Stabla odluke

Stabla odluke se, osim u strojnom učenju, konceptualno koriste i mnogo šire - naime, dijagrami stabla odluke su se pokazali jako korisni pri vizualizaciji i donošenju odluka u poslovnom svijetu [1].
Ovdje ćemo razmatrati stabla odluke kao metodu strojnog učenja. Kao i sve metode koje slijede, ova će se koristiti za predviđanje vrijednosti ciljnog atributa koristeći ulazne vrijednosti. Listovi stabla odluke predstavljaju moguće vrijednosti ciljnog atributa, koje kod problema klasifikacije predstavljaju razrede, a ostali čvorovi stabla predstavljaju ulazne atribute. Također, stabla odluke su najčešće upravo binarna stabla.
Pretpostavimo, jednostavnosti radi, da želimo stablom odluke napraviti klasifikaciju jednog ciljnog atributa za nekoliko ulaznih atributa. Tada će stablo odluke za svaki čvor koji nije list imati oznaku jednog od ulaznih atributa, odnosno neki uvjet grananja vezan za njegovu vrijednost. Svaki od tih čvorova se tada ovisno o ulaznoj vrijednosti grana u čvor ispod koji može biti list ili još jedan atributni čvor. Svaki list stabla je označen razredom klasifikacije ili distribucijom vjerojatnosti preko istih razreda, pa će se tako ulazni podatak klasificirati u razred ili u određenu distribuciju.
Stablo se gradi razdvajanjem izvornog skupa podataka, koji predstavlja korijen stabla, u podskupove - koji predstavljaju djecu tog korijena. Razdvajanje se radi na temelju pravila razdvajanja, koja se izvode iz atributa klasifikacije. Ovaj se proces ponavlja na svakom podskupu rekurzivno. Konačni uvjet rekurzije se definira tako da je rekurzija gotova kad podskup na nekom čvoru ima sve iste vrijednosti cijele varijable, ili kad razdvajanje više ne pridonosi predviđanju, odnosno klasifikaciji. Ovaj proces gradnje stabla od vrha prema dnu

(*top-down induction of decision trees* [2]) je primjer pohlepnog algoritma, te se takav pristup najčešće i koristi.



Slika 2.1: Primjer stabla odluke učenog nad skupom podataka o perunikama koristeći knjižnicu *scikit-learn*

Pogledajmo kako stablo sa slike 2.1 radi predikcije, uz ulazni podatak koji predstavlja dimenzije perunike. Počinjemo na korijenu stabla, gdje provjeravamo da li je duljina latice manja od 2.45 cm. Ako je, onda se pomičemo na lijevo dijete, koje je list, pa tada možemo klasificirati cvijet kao *Iris Setosa*. Inače se pomičemo na desno dijete u stablu, te provjeravamo širinu latice.

Atribut čvora *samples* nam kaže koliko instanci naučenog skupa pripada tvom čvoru. Primjerice, 100 instanci ima duljinu latice veću od 2.45 cm, od kojih 54 ima širinu latice manju od 1.75 cm. Atribut *value* pokazuje koliko instanci pripada kojem od razreda, gdje su u našem slučaju razredi redom *Iris Setosa, Iris Versicolor* i *Iris Virginica*. Konačno, svojstvo *gini* predstavlja „nečistoću" tog čvora, gdje je čvor „čist" (gini = 0) ako sve instance skupa za učenje pripadaju istom razredu. Jednadžba 2.1 pokazuje kako se računa ocjena *gini* za neki čvor i, gdje je $p_{i,k}$ omjer instanci razreda k u odnosu na broj instanci skupa za učenje i-tog čvora.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{2.1}$$

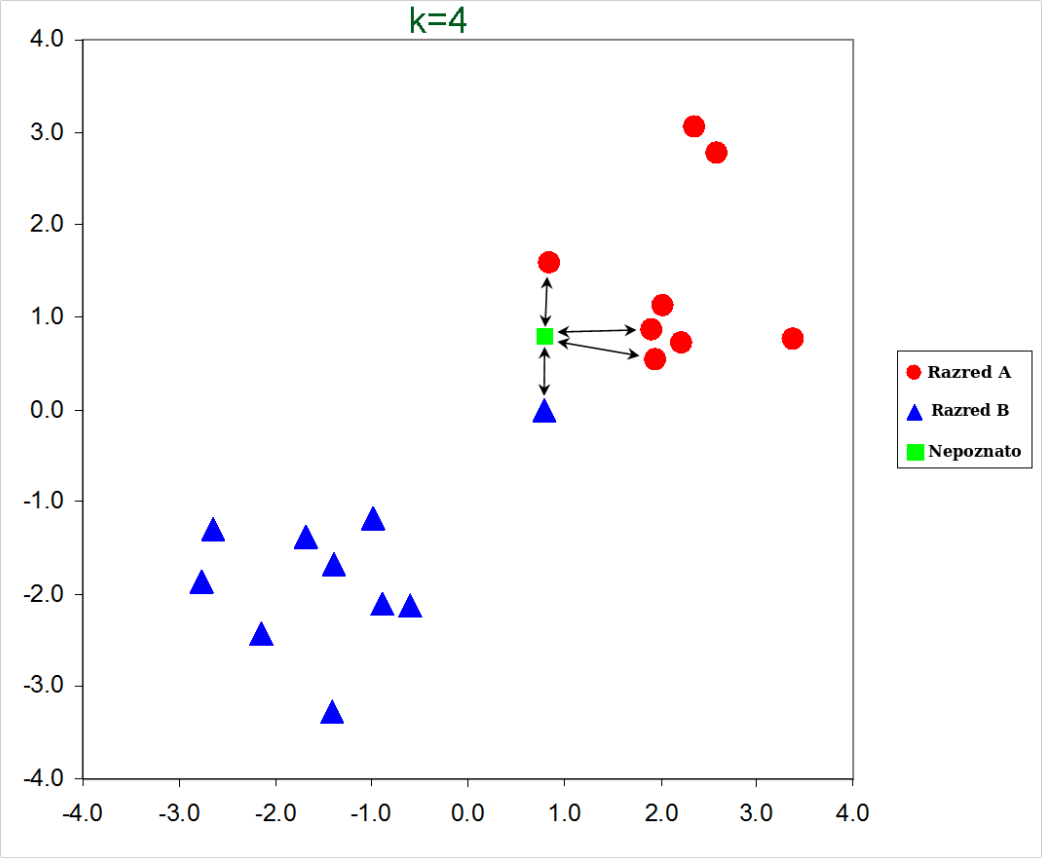
Često korišten algoritam za izgradnju stabala, zvan *Classification And Regression Tree* (CART) koristi upravo gore spomenutu ocjenu čistoće kako bi donio odluku kako razdvojiti podskupove. Ideja je jednostavna - algoritam najprije razdvoji skup za učenje u dva podskupa koristeći jedan atribut skupa k i prag t_k. Ove varijable algoritam odabere na način da pronađe par koji proizvodi najčišći podskup za danu veličinu podskupa. Funkcija cijene koju algoritam nastoji minimizirati je dana jednadžbom 2.2:

$$J\left(k, t_k\right) = \frac{m_{\text{ lijevo}}}{m} G_{\text{ lijevo}} + \frac{m_{\text{ desno,}}}{m} G_{\text{ desno}} \tag{2.2}$$

, gdje je $G_{\text{ lijevo/desno}}$ mjera nečistoće lijevog/desnog podskupa, a $m_{\text{ lijevo/desno}}$ broj instanci u lijevom/desnom podskupu.

2.2. Algoritam K-najbližih susjeda

Algoritam k-najbližih susjeda je klasifikacijska metoda strojnog učenja, i jedna od temeljnih i najjednostavnijih za razumijevanje. Smatra se najprikladnijom za korištenje kada ne postoji prethodno znanje o distribuciji skupa podataka nad kojim se izvodi prediktivna analiza. Ova se klasifikacijska metoda razvila iz potrebe za izvođenjem diskriminacijske analize kada su parametarske procjene ili gustoće vjerojatnosti nepoznate, ili teške za odrediti.



Slika 2.2: Jednostavan primjer klasifikacije nad 19 uzoraka. Od 4 najbližih susjeda testnog uzorka, najčešći razred je razred A, pa se stoga uzorak klasificira u taj razred.

Ovaj se klasifikator tipično temelji na Euklidovoj udaljenosti između testnog uzorka i specificiranih uzoraka za učenje. Neka je $\mathbf{x_i}$ ulazni uzorak s p svojstava $(x_{i1}, x_{i2}, ...x_{ip})$, n ukupan broj ulaznih uzoraka $(i = 1, 2, ..., n)$ i p ukupan broj svojstava $(j = 1, 2, ..., p)$. Euklidova udaljenost između $\mathbf{x_i}$ i $\mathbf{x_l}$ $(l = 1, 2, ..., n)$ je definirana kao:

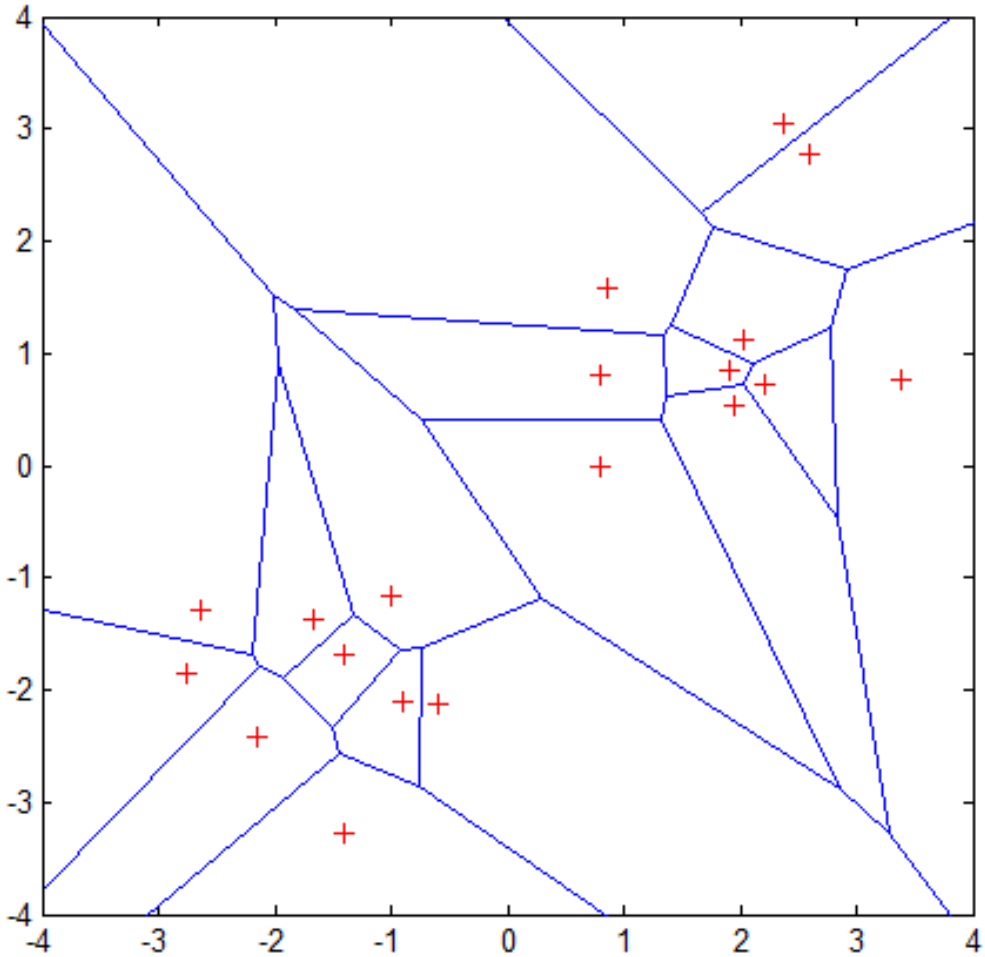
$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \cdots + (x_{ip} - x_{lp})^2} \tag{2.3}$$

Grafička predodžba koncepta najbližih susjeda je ilustrirana Voronojevim mozaikom [3] prikazanoj na slici 2.3. Teselacija pokazuje 19 uzoraka označenih s „+”-om, i Voronojevu ćeliju, R, koja okružuje svaki uzorak. Ta ćelija enkapsulira sve susjedne točke koje su najbliže

pojedinom uzorku, i definira se kao

$$R_i = \{\mathbf{x} \in \mathbb{R}^p : d(\mathbf{x}, \mathbf{x}_i) \leq d(\mathbf{x}, \mathbf{x}_m), \quad \forall \quad i \neq m\} \tag{2.4}$$

gdje je R_i Voronijeva ćelija za uzorak $\mathbf{x_i}$, te $\mathbf{x}1$ predstavlja sve moguće točke unutar ćelije R_i. Voronojevi mozaici odražavaju dvije karakteristike koordinatnog sustava: sve točke unutar ćelije nekog uzorka će biti najbliži susjed tog uzorka, te za svaki uzorak će najbliži susjed biti određen najbližem od ruba ćelije. Koristeći drugu karakteristiku, klasifikacijsko pravilo ovog algoritma bit će jednostavno dodjeljivanje prevladavajuće kategorije od k najbližih uzoraka učenja. U praksi se k najčešće uzima kao neparan broj, kako bi se izbjegla potencijalna izjednačenja. Ako vrijedi $k = 1$, tada promatramo pravilo klasifikacije poznatije kao *najbliži susjed*.



Slika 2.3: Voronojev mozaik koji pokazuje ćelije od 19 uzoraka.

2.3. Ansambli stabala odluke

Ansambli općenito koriste pristup „podijeli pa vladaj" koji se u domeni strojnog učenja koristi za poboljšanje performansi. Osnovni princip je da skupina slabijih modela zajedno može postići bolje karakteristike. Ansambli stabala odluke koriste više manjih stabala odluke istim principom. Prvi algoritam za ansamble stabala osmislio je Tin Kam Ho [4] koristeći metodu nasumičnih podskupova. Iako prije spomenuta stabla odluke imaju jasnu primjenu, ne dolaze bez svojih nedostataka. Primjerice, stabla koja se grade do velikih dubina imaju tendenciju naučiti znatno neregularne uzorke, što znači da mogu „prenaučiti" (engl. *overfitting*) skupove podataka nad kojima se učenje provodi. Ovo predstavlja problem prilikom klasifikacije novih uzoraka koji nisu bili dio skupa podataka za učenje. Ansambli stabala odluke ovaj problem rješavaju tako da uzimaju u obzir prosjek više dubokih stabala odluke, učenih na više različitih dijelova istog skupa za učenje, s ciljem smanjenja varijance. Cijena ovoga je gubitak interpretabilnosti, ali se performanse konačnog modela znatno povećavaju, što je Tin Kam Ho pokazao u svom inicijalnom članku.
Algoritam za treniranje primjenjuje tzv. pakiranje (engl. *bagging*) nad stablima. Uz skup podataka za treniranje $\mathbf{X} = x_1, ..., x_n$ s ciljnim atributima $\mathbf{Y} = y_1, ..., y_n$, opetovano pakiranje ($B$ puta) odabire nasumičan podskup od $\mathbf{X}$ te ga trenira na novom stablu odluke:

Algoritam

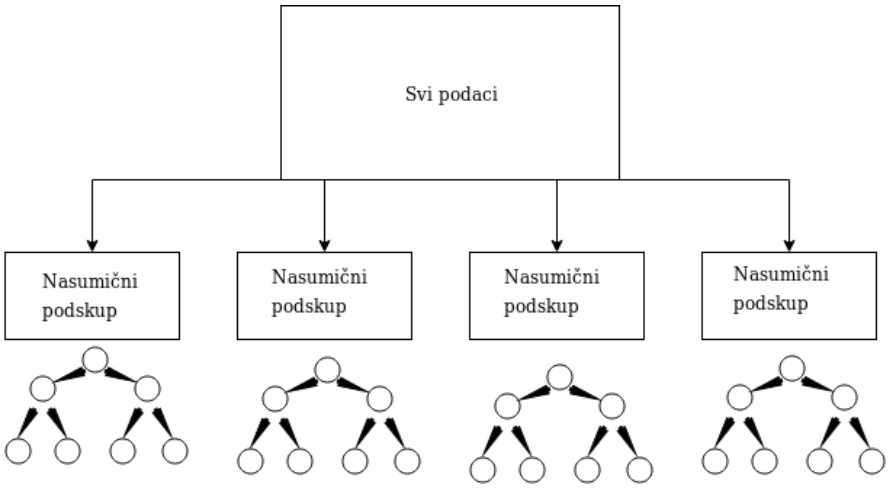
Za $b = 1, ..., B$:

1. Odaberi (s mogućnošću pojavljivanja istog uzorka u više podskupova) n uzoraka za učenje iz $\mathbf{X}$, $\mathbf{Y}$; nazovimo ih $\mathbf{X}_b$, $\mathbf{Y}_b$.
2. Uči stablo odluke f_b nad $\mathbf{X}_b$, $\mathbf{Y}_b$.

Nakon učenja, klasifikacija za nove uzorke x' mogu se napraviti uzimanjem najčešće pojavljivanog razreda predviđanja sa svih stabala odluke posebno, slično kao kod algoritma k-najbližih susjeda:

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x') \tag{2.5}$$

Broj uzoraka/stabala, B, je slobodan parametar. Tipično se koristi između nekoliko stotina do više tisuća stabala, ovisno o veličini i prirodi skupa za učenje. Optimalan B može se naći promatranjem prosječne pogreške predviđanja na svakom uzorku x_i, koristeći isključivo stabla koja nisu imala x_i u svom podskupu [5].



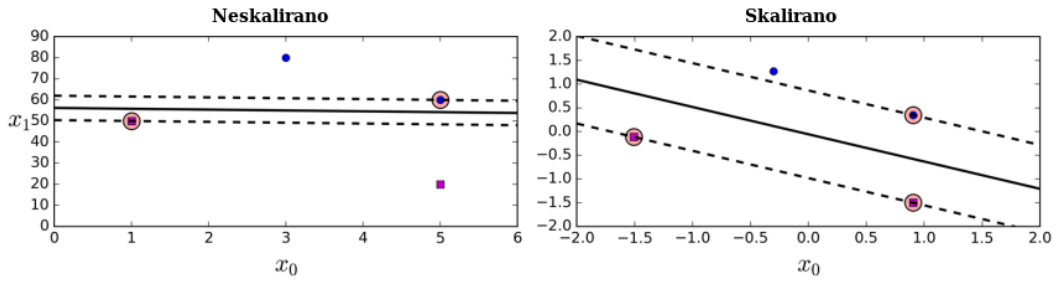
Slika 2.4: Dijagram koji prikazuje konceptualnu ideju izgradnje modela ansambla stabala odluke.

Gornja procedura opisuje originalni algoritam pakiranja koji se u početku koristio sa stablima odluke. Moderna implementacija, poznatija po engleskom nazivu *random forest*, koristi modificiranu verziju originalnog algoritma koja podskupove stvara nasumičnim biranjem *značajki* iz originalnog skupa. Razlog tom pristupu je činjenica da ako neke značajke više pridonose predviđanju konačnog rezultata, te će se značajke naći u većini od ukupno B stabala, uzrokujući korelaciju, što pridonosi performansama [6].

2.4. Strojevi s potpornim vektorima

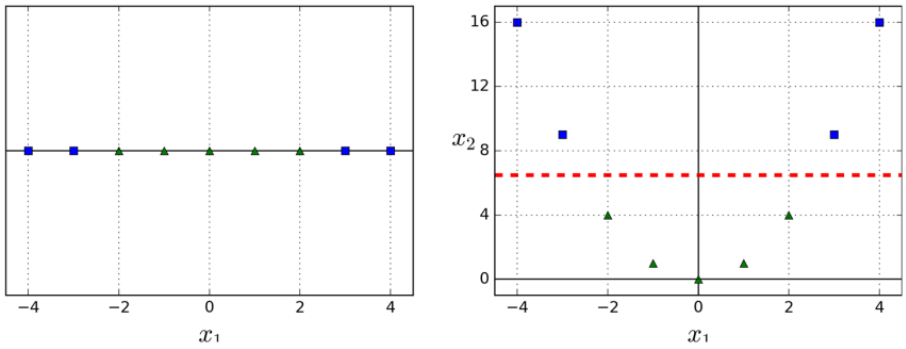
Cilj klasifikacije je particioniranje skupa podataka na temelju nekog kriterija kako bi se podaci organizirali u neki značajniji oblik. Postoji više načina kako bi se postigao taj cilj, a jedan od prirodnijih bi bio postavljanje granica u ona mjesta podatkovnog prostora gdje ima najmanje podataka. Ovaj način koristi metoda klasifikacije korištenjem strojeva s potpornim vektorima.

U suštini, cilj stroja s potpornim vektorima je pronaći hiperravninu u N-dimenzionalnom prostoru koja na najbolji način klasificira više podatkovnih uzoraka. Najbolja klasifikacija će se napraviti pronalaskom hiperravnine koja ima maksimalnu marginu, odnosno najvišu udaljenost od krajnjih podatkovnih uzoraka oba razreda. Maksimizacija te marginalne udaljenosti nam omogućava da s većom sigurnošću možemo klasificirati nove podatke koristeći naučeni model. Strojevi potpornih vektora su znatno osjetljiviji na skaliranje svojstava skupa podataka, te se na slici 2.5 može vidjeti kako skaliranje utječe na prije spomenute margine. Naime, na drugoj slici gdje je primjenjeno skaliranje vidimo kako linija koja razdvaja dva razreda klasifikacije ima veću udaljenost od krajnjih točaka podatkovnog skupa. Ovo znači da će dobiveni model s većom sigurnošću raditi procjene za nove podatke nego da nismo skalirali ulazne podatke.



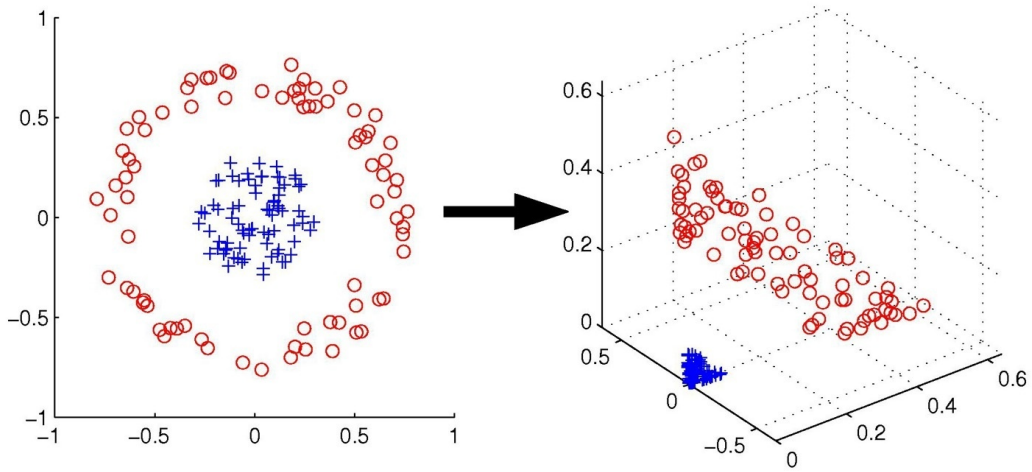
Slika 2.5: Razlika među marginama hiperravnina primjenom skaliranja skupa podataka.

Iako su linearni klasifikatori potpornim vektorima efikasni za neke slučajeve, velik broj skupova podataka iz stvarnog života nemoguće je linearno separirati. Jedan pristup kod takvih skupova je dodavanje još svojstava. Na slici 2.6 prikazan je jedan skup podataka sa samo jednim svojstvom x_1. Vidljivo je kako taj skup podataka nije linearno razdvojiv, ali dodavanjem drugog svojstva $x_2 = x_1^2$ se postiže razdvojivost u dvije dimenzije.



Slika 2.6: Postizanje separabilnosti skupa podataka dodavanjem novog svojstva.

Dodavanje polinomnih svojstava je jednostavno za implementaciju, ali niži stupanj polinoma ne može se nositi s jako složenim skupovima podataka, dok viši stupnjevi mogu stvoriti ogromne brojeve svojstava, što znatno usporava model. Srećom, otkriće jezgrenih metoda ponudilo je pojednostavljenje ovog problema kombinatorne eksplozije. Jezgrene metode omogućuju izvođenje operacija u višedimenzionalnom prostoru bez da se ikad računaju koordinate podataka u tim višim dimenzijama, već matričnim operacijama između parova podataka u dimenzionalnom prostoru koji odgovara originalnom broju svojstava, što je mnogo manje zahtjevno za računala. Na slici 2.7 prikazano je kako se može teško odvojiv skup podataka u nižim dimenzijama lako odvojiti adekvatnom hiperravninom u višoj dimenziji.

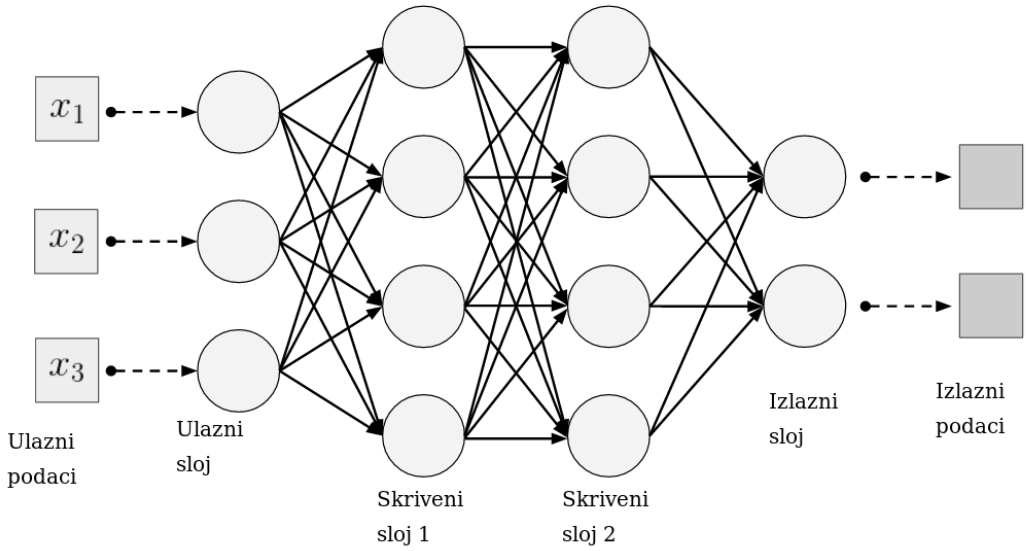


Slika 2.7: Postizanje separabilnosti skupa podataka hiperravninom prelaskom u višu dimenziju.

2.5. Neuronske mreže

Možda najjednostavniju definiciju neuronske mreže, konkretnije umjetne neuronske mreže, utemeljio je izumitelj jednog od prvih neuroračunala, dr. Robert Hecht-Nielsen. Njegova definicija glasi: „...računalni sustav sastavljen od određenog broja jednostavnih, višestruko povezanih elemenata koji obrađuju informacije dinamičkim odzivom stanja na ulazne podatke." [7]. Umjetne neuronske mreže su okvirno modelirane na temelju moždane kore sisavaca, iako u značajno manjem razmjeru u trenutku pisanja ovog rada. Veća umjetna neuronska mreža može imati nekoliko tisuća elemenata, dok ljudski mozak ima više milijardi međusobno povezanih neurona.

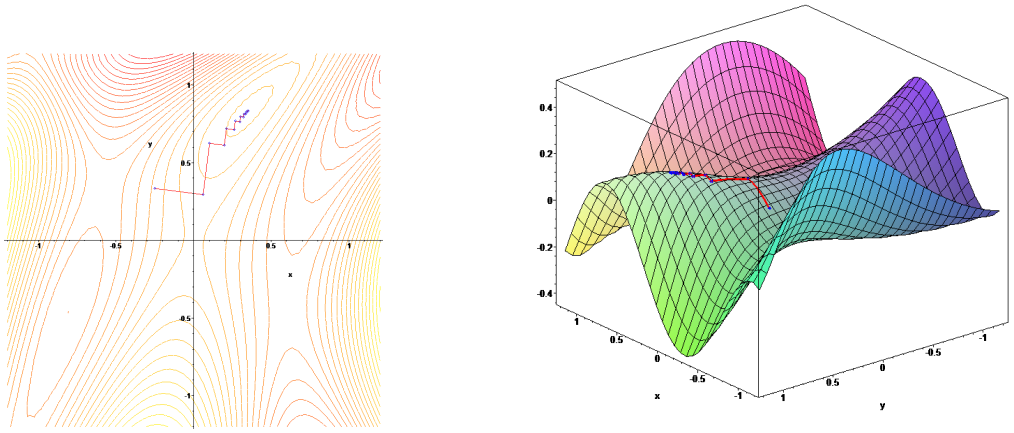
Iako matematička pozadina neuronskih mreža nije nimalo trivijalna, korisnik može dobiti jednostavno operativno razumijevanje njihove strukture i funkcije. Neuronske mreže su tipično organizirane slojevito, gdje se pojedini sloj sastoji od broja međusobno povezanih čvorova koji sadrže aktivacijsku funkciju. Obrasci na kojima mreža uči i radi predikcije dolaze na ulazni sloj mreže, koja komunicira s jednim ili više uzastopnih skrivenih slojeva, gdje se obrada podataka radi pomoću sustava povezanih težinskih funkcija. Skriveni slojevi su u konačnici povezani s izlaznim slojem koji daje rezultat predikcije.



Slika 2.8: Ilustrativni primjer jednostavne neuronske mreže.

Kako bi neuronske mreže mogle učiti svoj model, one koriste pravila učenja koja modificiraju težinske vrijednosti veza između neurona na temelju obrazaca u ulaznim podacima za učenje. Može se reći da neuronske mreže uče na sličan princip kao i biološki sustavi neurona; na primjer, dijete nauči prepoznavati predmete u stvarnom svijetu na temelju više primjera istog pojedinog predmeta.

Jedan od najpoznatijih pravila učenja je tzv. *delta pravilo*, koje se temelji na propagiranju pogreške predikcije unatrag. Ovdje je učenje nadziran proces gdje neuronska mreža u početku nasumično „pogađa" vrijednost predikcije. Zatim se provjerava razlika predviđene i stvarne vrijednosti, te naprave adekvatna podešavanja težinskih vrijednosti. Sama podešavanja se matematički provode gradijentnim spustom unutar vektorskog prostora rješenja putem najstrmijeg vektora na ravnini pogreške. Globalni minimum je rješenje s teoretski najmanjom pogreškom, no u većini slučajeva ravnina pogreške imat će mnogo različitih lokalnih minimuma koji neće nužno dati zadovoljavajuće rješenje. Stoga se cijeli proces učenja propagiranjem pogreške unatrag i gradijentnim spustom ponavlja u više *epoha*.



(a) Konturni prikaz plohe

(b) 3D prikaz plohe

Slika 2.9: Primjena gradijentnog spusta nad plohom $F(x, y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos\left(2x + 1 - e^y\right)$

3. Interpretabilnost i točnost metoda strojnog učenja

Da bismo usporedili metode strojnog učenja, potrebno je biti upoznat s načinima ocjenjivanja performansi istih metoda. U ovom poglavlju ćemo se stoga upoznati s procesima učenja i ocjenjivanja modela koji se u praksi pokazuju najefikasnijima. Također, kako bi došli do poveznice između interpretabilnosti i performansi, potrebno je definirati što je to interpretabilnost uopće u svijetu strojnog učenja, i zašto je važna.

3.1. Učenje i testiranje metoda

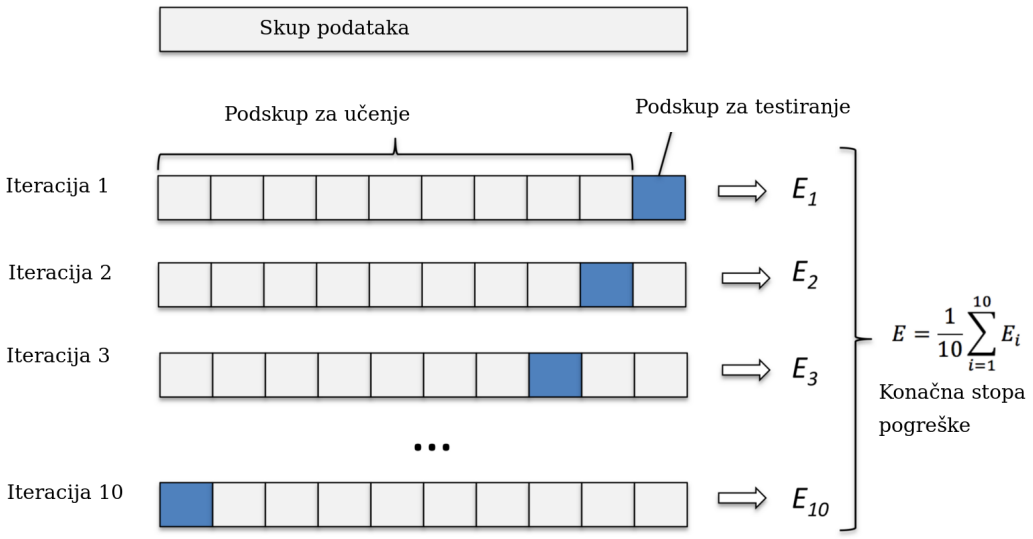
U suštini, strojno učenje je usredotočeno na poboljšanje sposobnosti sustava na način da isti sustavi uče od podataka, umjesto eksplicitnog definiranja pravila. Generalan proces u projektima baziranim na strojnom učenju jest skupljanje podataka o problemskoj domeni, priprema istih podataka kako bi bili adekvatni za učenje, te definiranje ciljnog atributa. Zatim je potrebno definirati da li je problem koji rješavamo klasifikacijske ili regresijske prirode, te koji je algoritam ili model najadekvatniji. Konačno, pripremljeni se podaci mogu poslati odabranom algoritmu koji će na temelju njih učiti.
Ako je algoritam temeljen na instancama, poput k-najbližih susjeda ili strojeva s potpornim vektorima, onda će zapamtiti ulazne podatke i koristiti neku mjeru sličnosti kako bi donosio zaključke nad novim podacima. S druge strane, ukoliko je algoritam temeljen na modelu, poput neuronskih mreža, on će u iteracijama namještati svoje parametre kako bi postignuo što veću točnost nad svojim podacima.
Ipak, jedini način kako bi sa sigurnošću utvrdili da li algoritam dobro radi predikcije jest da mu damo podatke nad kojima nije bio učen. Stoga je dobra praksa dostupne podatke najprije razdvojiti u skup za učenje i skup za testiranje. Na skupu za učenje će se učiti model kao i do sad, dok će se skup za testiranje iskoristiti kako bi utrvdili da li algoritam dobro generalizira nad novim podacima uz naučene parametre. Preporučeno je dvije trećine cjelokupnog skupa podataka iskoristiti za učenje, a trećinu za testiranje [8].

Stopa pogreške (omjer pogrešnih klasifikacija i ukupnog broja instanci) nad skupom za testiranje zove se stopa generalizacijske pogreške, i evaluacijom modela nad skupom za testiranje možemo dobiti tu vrijednost. Ona nam kaže upravo kako će se naš istrenirani model ponašati nad dosad neviđenim podacima. Ako model ima nisku stopu pogreške nad skupom za učenje, a visoku nad skupom za testiranje, to je dobar indikator da je model *prenaučen*.

3.2. Unakrsna validacija

Razmotrimo sada slučaj gdje je količina podataka za učenje i testiranje ograničena. Gore spomenuta metoda separacije skupa podataka može imati kao rezultat nereprezentativnost skupa za učenje (ili čak testiranje) u odnosu na cjelokupni skup, što može negativno utjecati na model.

Jedna generalna i često korištena metoda koja može pomoći (i koja će se koristiti u praktičnom dijelu ovog rada) jest *unakrsna validacija*. Općenit algoritam kaže da se cijeli proces učenja i testiranja modela ponavlja k puta, čime dobivamo k-struku unakrsnu validaciju. Podaci se podijele u k podskupova, najčešće uz stratifikaciju, koja omogućuje da razdioba vrijednosti ciljnog atributa ostane zadržana u svakom podskupu. Jedan od tih podskupova se koristi za testiranje, a prije toga se model uči na preostalim podacima. Zatim se uzimaju u obzir stope pogrešaka svih iteracija i računa se njihov prosjek.



Slika 3.1: Primjer unakrsne validacije za $k = 10$.

3.3. Interpretabilnost metoda strojnog učenja

Metode strojnog učenja se međusobno umnogome razlikuju po pitanju lakoće interpretacije. Iako postoji konkretna i objašnjiva matematička pozadina iza svakog od algoritama, problem leži u objašnjenju *zašto* je neki algoritam došao do zaključka do kojeg je došao. Ako je neuronska mreža detektirala da se određena osoba pojavljuje na nekoj slici, postavlja se pitanje što je na slici točno pridonijelo toj predikciji (oči, crte lica, nos...).
Adekvatna definicija interpretabilnosti u kontekstu strojnog učenja bila bi sposobnost objašnjavanja i predstavljanja nekog koncepta *na način razumljiv čovjeku*.
Postoji mnogo razloga zašto je interpretabilnost važan čimbenik u strojnom učenju. Jedan od relevannijih je deklamacija 71 [9] poznate Opće uredbe o zaštiti podataka Europske Unije (GDPR) koja koja kaže da svaki pojedinac ima pravo tražiti objašnjenje zašto je neki algoritam donio određenu odluku, ukoliko ona može utjecati na istog pojedinca, posebice pravno ili financijski. Nadalje, istraživači iz Googlea i Harvarda, Doshi-Velez i Kim u svom radu koji se bavi interpretabilnošću [10] navode više razloga zašto je ona bitna:

1. **Znanstveno razumijevanje**: čovjekov cilj je sticanje znanja, a budući da nemamo potpunu definiciju što znanje jest, najbolji pristup je tražiti objašnjenja koja se mogu pretvoriti u znanje.

2. **Sigurnost**: za složene zadatke, model strojnog učenja je gotovo nemoguće u potpunosti ispitati.

3. **Etika**: poželjno je zaštititi sustav od pojedinih oblika diskriminacije koji mogu utjecati na konačnu odluku.

4. **Kompromisi kao uzrok više ciljeva**: ako je algoritam namijenjen za više ciljeva, moguće je da će se doći do kolizije između njih; tipični primjeri su kolizija između privatnosti i prevencije diskriminacije, ili privatnosti i kvalitete predikcije.

U prije spomenutom radu, Dosh-Velez i Kim također daju zanimljive prijedloge za vrednovanje interpretabilnosti:

1. **Evaluacija osnovana na primjeni**: izvođenje eksperimenata na ljudima primjenjujući modele u cijelosti

2. **Metrike osnovane na ljudima**: izvođenje jednostavnijih eksperimenata na ljudima gdje bi cjeloviti eksperimenti bili teže izvedivi, ili neetički

3. **Evaluacija osnovana na funkciji**: nema potrebe za eksperimente na ljudima - primjenjivo kad se strojno učenje koristi za rješavanje ili optimizaciju problema koji su već validirani od strane ljudi u prijašnjim eksperimentima

Ryan Turner iz korporacije Northrop Grumman u svom članku [11] predlaže sustav evaluacije algoritama koji bi objasnio izlaz klasifikatora s nejasnim tumačenjem modela. Ključan aspekt predloženog sustava jest pružanje objašnjenja pojedinog izlaza klasifikatora, a ne skupa parametara. Cilj nije objasniti kako model generalno funkcionira, već samo pojedine slučajeve. Iako je razlika suptilna, takvo je objašnjenje mnogo jednostavnije od objašnjavanja cijelog modela. Stoga je krajnji cilj poboljšanje modela bez utjecaja na preciznost.

Iz priloženog vidimo da je u znanstvenom svijetu interes za interpretabilnošću sve veći, te postoji više načina kako se ona može povećati. Znanstvenici poput Ryana Turnera čak ne smatraju kako postoji kompromis između interpretabilnosti i performansi. Sposobnost tumačenja nije važna samo zbog moralnih načela i utjecaja kojeg strojno učenje ima na život čovjeka, već ona sama može pomoći da modeli postignu bolje performanse. Naime, ako znamo zašto je klasifikator negdje pogriješio, lakše ćemo kalibrirati parametre i/ili podatke kako bi sljedeći put rezultati bili bolji. Stoga možemo zaključiti kako veća interpretabilnost ne mora nužno uvjetovati manje performanse, te kako težnja boljem razumijevanju modela može samo pomoći.

4. Vrednovanje algoritama strojnog učenja koristeći Scikit-Learn

U praktičnom dijelu ovog rada cilj je bio napraviti aplikaciju koja ima mogućnost učitavanja skupa podataka i evaluaciju algoritma strojnog učenja nad istim skupom, kao što je opisano u poglavljima 3.1 i 3.2.

4.1. Knjižnica scikit-learn

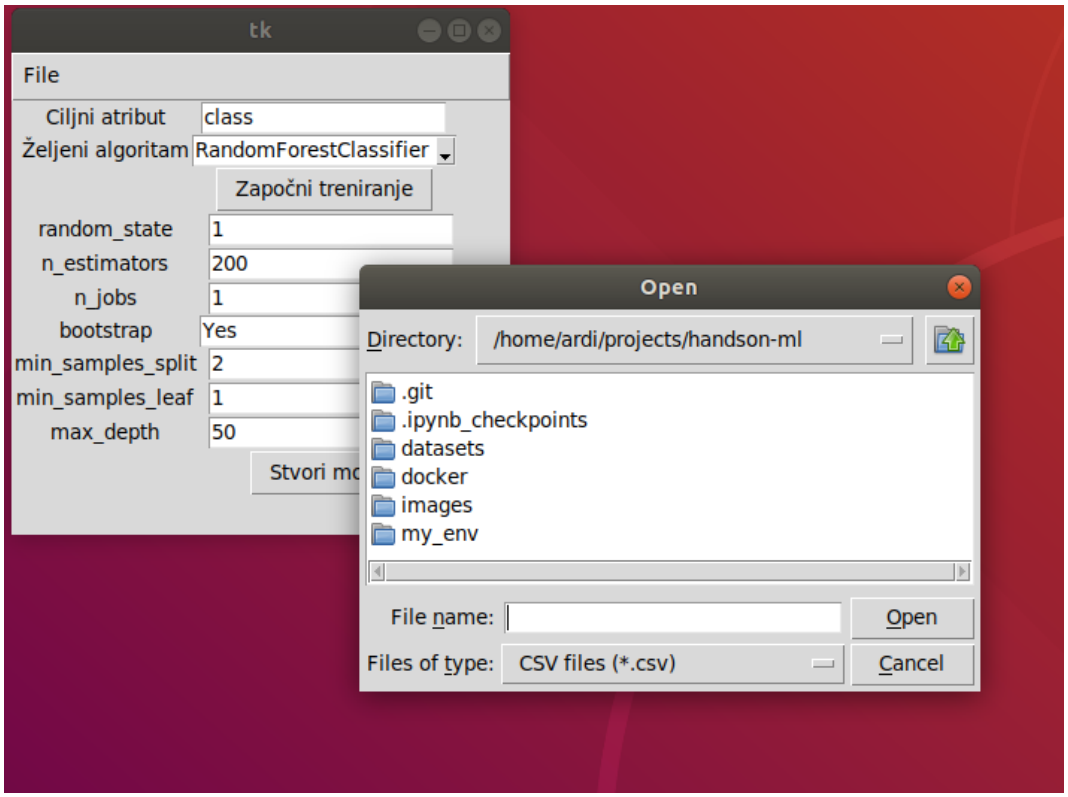
Scikit-learn [12] je besplatna knjižnica za strojno učenje za programski jezik Python. Projekt je započeo David Cournapeau 2007. godine kao projekt za Googleov *Summer of Code*, godišnji međunarodni program otvorenog koda. Matthieu Brucher se kasnije te godine priključio te radio na projektu kao temi svog doktorskog rada. Kasnije su Gael Varoquaux, Alexandre Gramfort i Vincent Michel preuzeli vodstvo projekta i izdali prvu javnu verziju, 2. veljače 2010. Od tada, nove inačice knjižnice su dolazile u tromjesečnim ciklusima, i veoma uspješna međunarodna *open-source* zajednica je vodila razvoj. Projekt je većim dijelom pisan u Pythonu, s nekim ključnim algoritmima napisanim u Cythonu radi boljih performansi.

Knjižnica je sama po sebi veoma moćna, te vrlo dobro dokumentirana na službenim stranicama. Nudi razne gotove algoritme strojnog učenja, te mnoge pomoćne alate za manipulaciju podacima poput skaliranja, stratifikacije i sličnih koji su vrlo korisni u područjima strojnog učenja i znanosti o podacima. Svakom se algoritmu mogu modificirati hiperparametri, te ga se može naučiti i testirati nad skupovima podataka. Uz to, podržane su razne metrike za validaciju i ocjenjivanje performansi algoritama.

Slika 4.1: Službeni logo knjižnice scikit-learn.

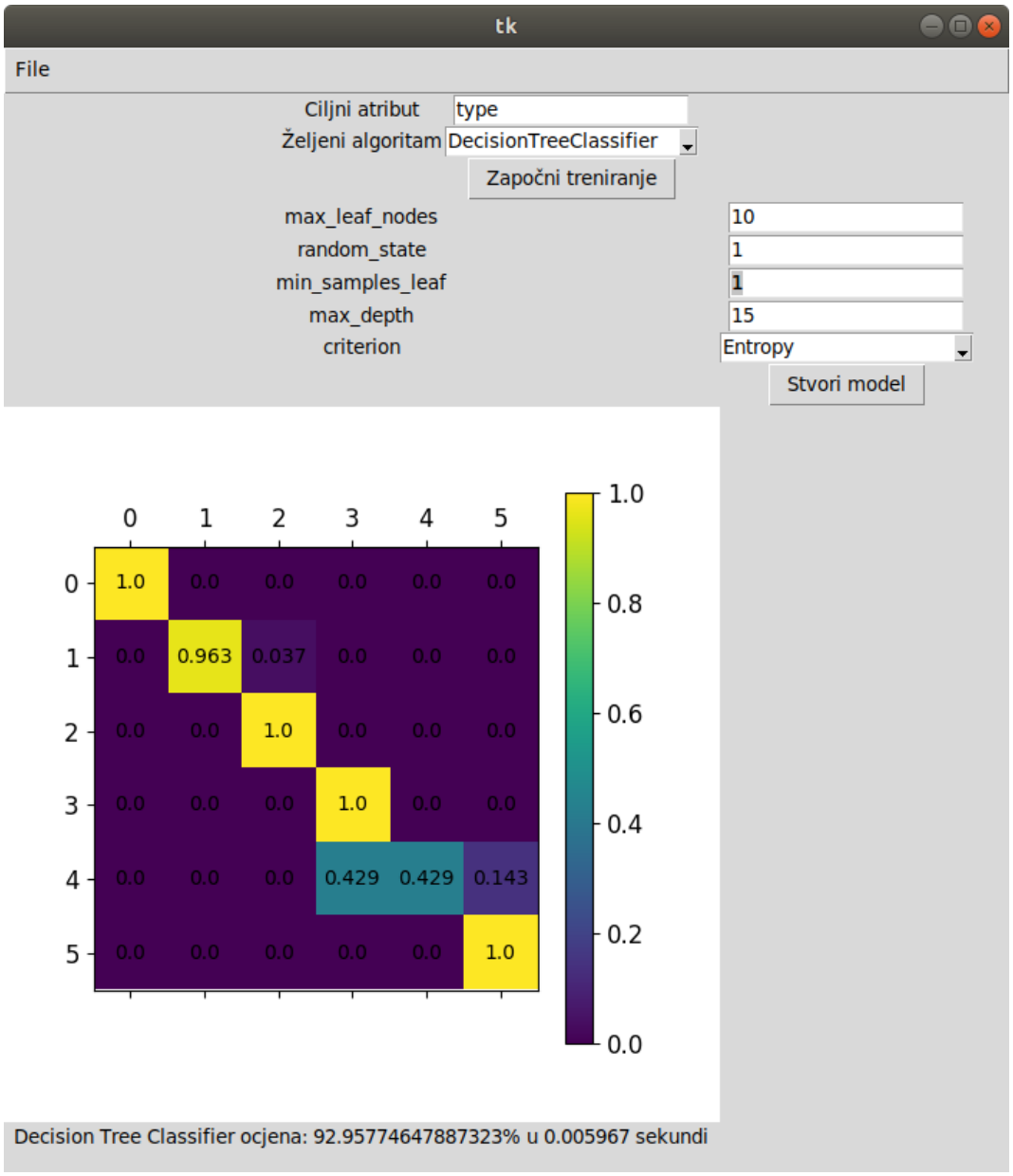
4.2. Aplikacija za učitavanje podataka, učenje i testiranje

Aplikacija se temelji na jednostavnom grafičkom korisničkom sučelju napravljenom koristeći biblioteku Tkinter. Tkinter omogućuje jednostavnu izradu korisničkog sučelja i nudi razne grafičke komponente koje možemo očekivati u tipičnoj stolnoj (engl. *desktop*) aplikaciji, poput biblioteke Swing za Javu ili Qt za C++. Scikit-learn, koji je spomenut u prethodnom odjeljku, je korišten za njegove algoritme strojnog učenja te metode validacije i metrike. Također se koristi biblioteka Pandas za učitavanje podataka.
Korištenje aplikacije je jednostavno. Najprije se učitaju podaci, za koje se očekuje da su u formatu CSV (Comma Separated Value), te se naznačuje ime ciljnog atributa klasifikacije. Zatim se odabire željeni algoritam strojnog učenja, te njegovi hiperparametri.



Slika 4.2: Odabir modela, njegovih hiperparametara i željenog skupa podataka.

Nakon odabira algoritma i hiperparametara klikom na Stvori model te Započni treniranje kreće evaluacija. Ispisat će se preciznost algoritma koristeći unakrsnu validaciju, te matrica konfuzije za sve razrede klasifikacije. Svaki redak matrice predstavlja predviđene vrijednosti, a stupci označene (stvarne), gdje je $C_{i,j}$ broj klasifikacija koje pripadaju razredu i a klasificirane su u razred j. Na slici 4.3 matrica je normalizirana.



Slika 4.3: Treniranje modela stabla odluke sa specifičnim parametrima na skupu podataka, vizualizacija i ispis rezultata.

4.3. Eksperimentalni rezultati vrednovanja algoritama

U ovom odjeljku primjenjuju se svi algoritmi strojnog učenja iz poglavlja 2 na skupovima podataka iz stvarnog svijeta. Za sve podatkovne skupove korišteni su algoritmi knjižnice scikit-learn, s istim hiperparametrima:

1. **Stablo odluke**: kriterij čistoće - *gini*, razdvajanje - po najboljoj ocjeni, maksimalna dubina $= \infty$, minimalan broj uzoraka za razdvajanje $= 2$, minimalan broj uzoraka za list $= 1$

2. **K-najbližih susjeda**: $K = 19$, metrika udaljenosti - Manhattanova udaljenost, težinske vrijednosti susjeda - uniformne

3. **Stroj s potpornim vektorima**:$C = 10$, $\gamma = 0.001$, maksimalno iteracija $= \infty$

4. **Ansambl stabala odluke**: broj stabala $= 100$, kriterij čistoće - *gini*, maksimalna dubina $= \infty$, minimalan broj uzoraka za razdvajanje $= 2$, minimalan broj uzoraka za list $= 1$

5. **Neuronska mreža**: aktivacijska funkcija - ReLU ($f(x) = x^+$), algoritam podešavanja težinskih vrijednosti - stohastički gradijentni spust, maksimalni broj iteracija $= 500$

Skupovi podataka su dohvaćeni sa službene stranice Sveučilišta u Kaliforniji [13]. Točnost je definirana kao prosječna od svih dobivenih ocjena točnosti dobivenih k-strukom unakrsnom validacijom, dok je tolerancija izračunata kao standardna devijacija istih ocjena.
Učenje i testiranje algoritama izvedeno je na Intelovom procesoru za prijenosna računala *Intel Core i7-4700MQ*.

Breast cancer, Wisconsin [14], 569 uzoraka, 2 razreda, 32 atributa			
Algoritam	Točnost (%)	Tolerancija (%)	Vrijeme izvođenja [s]
Stablo odluke	92.3752	4.7804	0.0456
K-najbližih susjeda	92.6451	3.2957	0.0239
Stroj s potpornim vektorima	91.8556	3.9923	0.0907
Ansambl stabala odluke	95.7962	2.9341	0.9349
Neuronska mreža	81.1134	16.7021	2.6487

Iris [15], 150 uzoraka, 3 razreda, 4 atributa			
Algoritam	Točnost (%)	Tolerancija (%)	Vrijeme izvođenja [s]
Stablo odluke	94.0000	6.6333	0.0093
K-najbližih susjeda	94.0000	6.6333	0.0157
Stroj s potpornim vektorima	92.0000	6.0000	0.0124
Ansambl stabala odluke	95.0000	6.7082	0.5834
Neuronska mreža	98.0000	4.0000	2.5320

Digits [16], 5620 uzoraka, 10 razreda, 64 atributa			
Algoritam	Točnost (%)	Tolerancija (%)	Vrijeme izvođenja [s]
Stablo odluke	82.4642	3.4411	0.1095
K-najbližih susjeda	95.9263	2.3972	0.1932
Stroj s potpornim vektorima	99.0861	0.9431	0.8004
Ansambl stabala odluke	97.3402	1.9960	1.8666
Neuronska mreža	96.0937	1.7037	25.5989

Ionosphere [17], 351 uzoraka, 2 razreda, 34 atributa			
Algoritam	Točnost (%)	Tolerancija (%)	Vrijeme izvođenja [s]
Stablo odluke	87.1558	6.7178	0.0573
K-najbližih susjeda	82.5725	6.2002	0.04212
Stroj s potpornim vektorima	85.1449	7.2196	0.0501
Ansambl stabala odluke	93.6413	2.8257	0.8703
Neuronska mreža	89.3297	7.4788	19.5717

Glass [18] 214 uzoraka, 7 razreda, 10 atributa			
Algoritam	Točnost (%)	Tolerancija (%)	Vrijeme izvođenja [s]
Stablo odluke	95.8095	6.5076	0.0329
K-najbližih susjeda	89.6190	11.098	0.0368
Stroj s potpornim vektorima	97.2381	4.6549	0.0320
Ansambl stabala odluke	95.8095	5.5525	0.676
Neuronska mreža	98.6190	2.7639	3.1735

Mammography [19], 961 uzoraka, 2 razreda, 6 atributa			
Algoritam	Točnost (%)	Tolerancija (%)	Vrijeme izvođenja [s]
Stablo odluke	75.1226	5.3166	0.0187
K-najbližih susjeda	79.3365	4.9231	0.0357
Stroj s potpornim vektorima	81.6707	6.3754	0.0917
Ansambl stabala odluke	77.4639	5.7929	0.817
Neuronska mreža	71.8726	8.6729	3.0113

Iz rezultata se vidi kako skup podataka nad kojim algoritam uči ima velik utjecaj na konačne performanse algoritma. Uz to, valja primjetiti kako iako kompleksniji algoritmi postižu generalno bolje rezultate, postoje slučajevi gdje se jednostavniji poput stabala odluke pokazuju kao validan izbor. Na jednostavnijim podatkovnim skupovima poput Iris-a jednostavniji algoritmi postižu jednake, a ponekad i bolje rezultate od kompleksnih.
Također je vidljivo kako kompleksniji algoritmi imaju generalno dulje vrijeme izvođenja na istim podatkovnim skupovima, uz očito isticanje neuronskih mreža. Valja napomenuti kako su korišteni podatkovni skupovi relativno mali, te se za složene probleme poput prepoznavanja obrazaca u slikama korisnost neuronskih mreža i sličnih algoritama mnogo više ističe, dok su npr. stabla odluke za takve probleme gotovo neupotrebljiva.

5. Zaključak

Algoritmi strojnog učenja moćan su alat za rješavanje problema klasifikacije većih skupova podataka. Svaki algoritam ima točno određene parametre i objašnjiva je pozadina iza njih, ali svi algoritmi nemaju jednako istaknutu interpretabilnost modela. Algoritmi poput neuronskih mreža sve su bolji za rješavanje kompleksnih problema poput prepoznavanja govora i slika, ali i sam model je kompleksan do te razine da ne možemo sa sigurnošću reći kako je točno donešena određena odluka. Kako strojno učenje napreduje velikom brzinom, ono postaje sve veći i veći faktor naših života pa je stoga važno imati solidno razumijevanje o njemu. Viša razina razumijevanja ovih algoritama i modela nam također omogućava da ih efikasnije koristimo, umjesto da ih tretiramo kao tzv. crne kutije. Poznati *Teorem nema-besplatnog-ručka* (engl. *No free lunch theorem*) kaže kako nijedan algoritam nije najbolje primjenjiv na sve probleme, što nam kaže da valja obratiti pažnju na to koji ćemo koristiti.

LITERATURA

[1] Mind Tools Content Team. Decision tree analysis. URL https://www.mindtools.com/dectree.html.

[2] J. R. Quinlan. Induction of decision trees. *Machine Learning, vol. 1, issue 1, pp. 81-106*, 1986.

[3] Georgy Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik, pp.97-178*, 1908.

[4] Tin Kam Ho. Random decision forests. *ICDAR '95 Proceedings of the Third International Conference on Document Analysis and Recognition - vol. 1, p. 278*, 1995. URL https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf.

[5] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani. *An Introduction to Statistical Learning.* Springer, 2013.

[6] Tin Kam Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis Applications, Volume 5, Issue 2, pp 102-112*, 2002.

[7] Maureen Caudill. *Neural Network Primer: Part 1.* Miller Freeman Publications, 1989.

[8] Witten, Frank, Hall, and Pal. *Data Mining, Practical Machine Learning Tools and Techniques, Fourth Edition.* Morgan Kaufmann, 2017.

[9] Europska Unija. General data protection regulation, recital 71. 2016. URL https://www.privacy-regulation.eu/en/r71.htm.

[10] Been Kim and Finale Doshi-Velez. Towards a rigorous science of interpretable machine learning. *Google AI*, 2017.

[11] Ryan Turner. A model explanation system. *Northrop Grumman Corporation*, 2015. URL http://www.blackboxworkshop.org/pdf/Turner2015_MES.pdf.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] D. J. Asuncion, A. i Newman. Uci machine learning repository. 2007. URL `http://www.ics.uci.edu/Bmlearn/MLRepository.html`.

[14] W.N. Street, W.H. Wolberg, and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. *Science and Technology, volume 1905, pp. 861-870*, 1993.

[15] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 1936.

[16] C. Kaynak. Methods of combining multiple classifiers and their applications to handwritten digit recognition. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, no. 3*, 1995.

[17] V G Sigillito, S P Wing, L V Hutton, and K B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Tech. Dig*, vol. 10: 262–266, 1989.

[18] I W Evett and E J Spiehler. Rule induction in forensic science. *Central Research Establishment, Home Office Forensic Science Service*, 1987.

[19] M. Elter, R. Schulz-Wendtland, and T. Wittenberg. The prediction of breast cancer biopsy outcomes using two cad approaches that both emphasize an intelligible decision process. *Medical Physics 34*, 2007.

Usporedba metoda strojnog učenja s jasnim i nejasnim tumačenjem modela

Sažetak

Različiti algoritmi klasifikacije, osim po samom pristupu problemu i performansama, razlikuju se i po razini razumljivosti i jednostavnosti objašnjenja čovjeku. Oni se mogu formalno usporediti po preciznosti kvalifikacije nad skupovima podataka koristeći metode poput unakrsne validacije, dok za interpretabilnost formalna definicija ne postoji. Ipak, sposobnost shvaćanja kako algoritmi donose odluke je sve bitniji s porastom odgovornosti sustava koji te algoritme koriste. Iako je generalan konsenzus da kompleksniji i teže razumljivi algoritmi donose bolje rezultate, oni jednostavniji imaju svoje prednosti i primjenjivi su na određene probleme.

Ključne riječi: Strojno učenje, klasifikacija, nadzirano učenje, algoritmi, interpretabilnost.

Comparison of Machine Learning Methods with Clear and Unclear Model Interpretation

Abstract

Different classification algorithms differ by their problem approach and performances, but also by their level of interpretability and ease of explanation. They can formally be compared using various methods such as cross validation, while no definition for interpretability exists. However, the ability to clearly explain how an algorithm works is growing in importance with the rapid growth of the algorithms' responsibilities. Although the general concensus is that more complex algorithms perform better, there are cases where the simpler models can prove more capable.

Keywords: Machine learning, classification, supervised learning, interpretability.