Istraživanje temperatura taljenja i kristalnih struktura metala primjenom strojnog učenja

Jelić, Borna

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Borna Jelić

ISTRAŽIVANJE TEMPERATURA TALJENJA I KRISTALNIH STRUKTURA METALA PRIMJENOM STROJNOG UČENJA

Diplomski rad

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA I INFORMATIKA; SMJER NASTAVNIČKI

Borna Jelić

Diplomski rad

Istraživanje temperatura taljenja i kristalnih struktura metala primjenom strojnog učenja

Voditelj diplomskog rada: izv. prof. dr. sc. Goranka Bilalbegović

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2021.

Zahvaljujem se profesorici Bilalbegović na velikoj pomoći i strpljenju.

Sažetak

Strojno učenje postaje sve značajnije za znanost i gospodarstvo. U ovom diplomskom radu se za analizu osobina elementarnih metala koriste metode nadziranog strojnog učenja linearna regresija i neuronske mreže. Metodom linearne regresije analizirane su korelacije temperatura taljenja za Youngove module elastičnosti, konstante rešetke i koeficijente linearne termičke ekspanzije. Primjenom neuronskih mreža ispitana je mogućnost predviđanja strukture kristalne rešetke metala. U metodičkom poglavlju predstavljeno je upoznavanje sa strojnim učenjem za učenike četvrtog razreda srednjih škola i to za k-sredine, algoritam nenadziranog strojnog učenja. Korišten je programski jezik Python i njegove biblioteke za strojno učenje.

Ključni pojmovi: fizika materijala, metali, nadzirano strojno učenje, linerana regresija, neuronske mreže, strojno učenje u srednjim školama, nenadzirano strojno učenje, algoritam k-sredina

Study of melting temperatures and crystal structures of metals using machine learning

Abstract

Machine learning is becoming increasingly important for science and economics. In this diploma thesis, the methods of supervised machine learning, linear regression and neural networks, are used to analyze the properties of elemental metals. The correlations of melting temperatures for Young 's modulus of elasticity, lattice constants and coefficients of linear thermal expansion were analyzed by the linear regression method. The possibility of predicting the structure of the metal crystal lattice was investigated using neural networks. The methodical section presents the introduction to machine learning for fourth grade high school students using k-means, an algorithm of unsupervised machine learning. The Python programming language and its machine learning libraries were used.

Keywords: materials physics, metals, supervised machine learning, linear regression, neural networks, machine learning in high schools, unsupervised learning, algorithm k-means

Sadržaj

1 Uvod

Fizika i kemija materijala su vrlo važne za razvoj društva. Materijale koristimo u svim područjima svakodnevnog života. Razvoj društva možemo pratiti preko materijala koji su se koristili kroz povijest. S vremenom su ljudi naučili proizvoditi i koristiti materijale koji se ne nalaze u prirodi. Svakog dana se u laboratorijima velikog broja znanstvenika istražuju materijali. Metali su jedna od najvažnijih grupa materijala. To mogu biti elementarni metali (kao što su aluminij, željezo, bakar, zlato, ili titanij), ali i njihovi spojevi od dva ili više metala, ili od metala i nemetala. U usporedbi s drugim materijalima, metali imaju veliku gustoću i čvrstoću, dobri su električni i toplinski vodiči. Nisu transparentni za svijetlost iz vidljivog područja elektromagnetkog spektra. Neki od metala (kao što su željezo, kobalt i nikal) pokazuju magnetske osobine. Sve te osobine su vrlo važne za primjene.

Brzi razvoj računalne znanosti bitno utječe na današnje metode istraživanja u svim znanstvenim područjima. Strojno učenje se razvija kao podskup istraživanja umjetne inteligencije [1]. Zasniva se na efikasnom procesiranju i analizi podataka i ima sve značajniju ulogu u znanosti i gospodarstvu. Koristeći algoritme strojnog učenja računala nam danas predlažu knjige i filmove, izbacuju spam poruke iz naše računalne pošte, klasificiraju fotografije objekata i ljudskih lica. Strojno učenje se sve više koristi i u znanosti o materijalima [2, 3].

U ovom radu se primjenjuju linearni regresijski model i neuronska mreža za analizu podataka o elementarnim metalima. Podatci se uzimaju iz baza Pymatgen (Python Materials Genomics) [4, 5] i Mendeleev [6]. Te baze se koriste u programima koji se izvršavaju na portalu NanoHub [7].

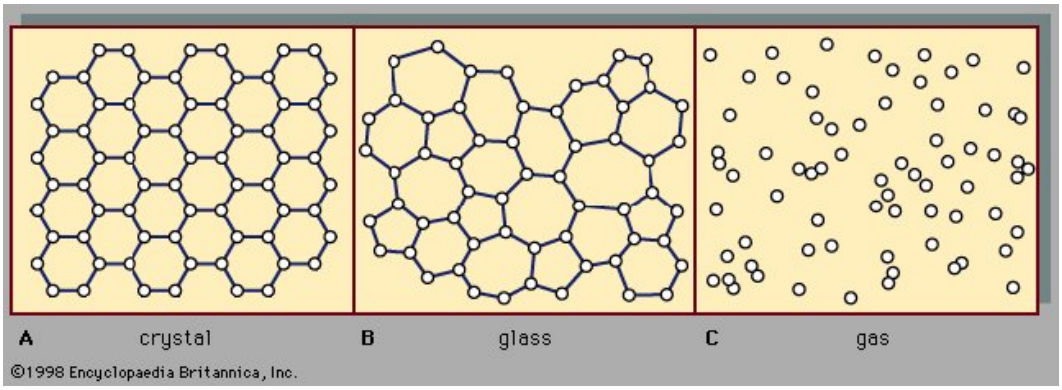
U drugom poglavlju se opisuju kristalne strukture. Specijalno su istaknute kubične i heksagonske rešetke zbog toga sto metali, čije se osobine analiziraju u četvrtom poglavlju, imaju takve strukture. U trećem poglavlju se predstavljaju osnovne ideje i algoritmi nadziranog strojnog učenja koji se koriste u ovom radu. U četvrtom poglavlju se navode korišteni programski paketi, opisuju i analiziraju rezultati. U metodičkom poglavlju su izloženi koncepti predstavljanja strojnog učenja u srednjim školama, pri čemu se koristi algoritam nenadziranog strojnog učenja k-sredine.

2 Struktura kristala

2.1 Kristali, amorfna tijela i tekućine

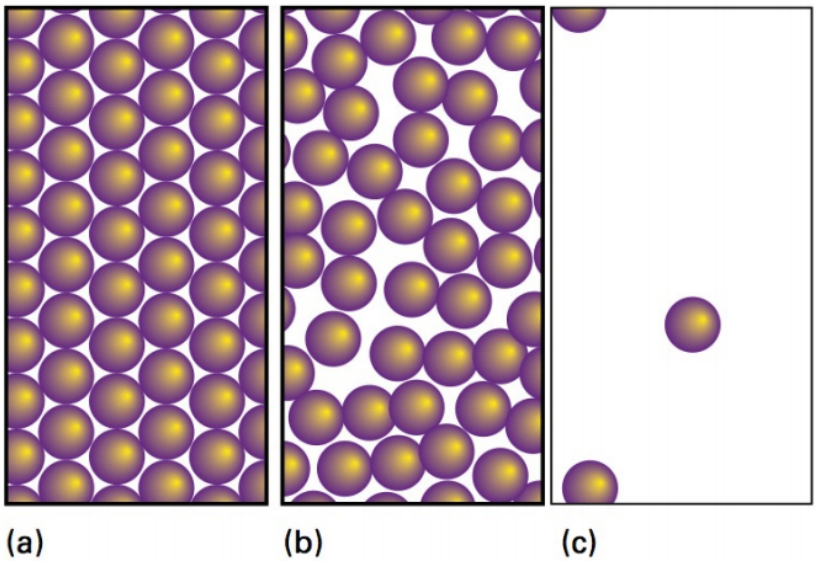
Za fiziku čvrstog stanja su vrlo važne kristalne strukture. Detalji strukture kristala su istraženi nakon otkrića rendgenskih zraka [8].

Postoje kristalna i amorfna čvrsta tijela. Razlika tih dviju klasa krutina je u njihovoj uredjenosti. Kod kristala postoji svojstvo koje zovemo *translacijska periodičnost (engl. translational periodicity)*, gdje se simetrični uzorci ponavljaju. Specijalnu vrstu uredjenosti imaju kvazikristali kod kojih nema translacijske uredjenosti, ali postoji diskretni difrakcijski dijagram. Kod amorfnih tijela ne postoji translacijska periodičnost. Primjer amorfnih tijela su stakla. Bitno je naglasiti da i u amorfnim tijelima postoji određeni nivo geometrijske uređenosti, za razliku od plinova gdje su atomi kaotično raspoređeni u prostoru [9]. Primjeri uređenosti u kristalu, staklu i plinu su prikazani na Slici 2.1.



Slika 2.1: Uređenje atomske strukture u kristalu (A), amorfnom tijelu (B) i plinu (C) [9]

Taljenje je fizikalni proces koji predstavlja prijelaz krutog stanja u tekuće dovođenjem topline [10]. Amorfne tvari se tale postupnim smanjenjem viskoznosti s povećanjem temperature. Unutarnja energija krutine se povećava, a kao posljedica i njena temperatura raste sve do temperature taljenja. Temperatura taljenja je ona temperatura pri kojoj krutine i tekućine mogu postojati u ravnoteži. Daljnim dovođenjem topline kruto stanje se pretvara u tekuće bez temperaturne promjene. Kada se čitava krutina otopi, dodatna toplina će povećati temperaturu tekućine. Temperatura taljenja je jedna od temeljnih osobina krutih tvari. Razlike u strukturi kristala, tekućine i plina te prijelazi između njih su prikazani na Slici 2.2.



Slika 2.2: Krutina prikazana na Slici (a) taljenjem prelazi u tekućinu na Slici (b). Isparavanjem tekućina prelazi u plin (Slika (c)) [10].

2.2 *Kristalne ćelije*

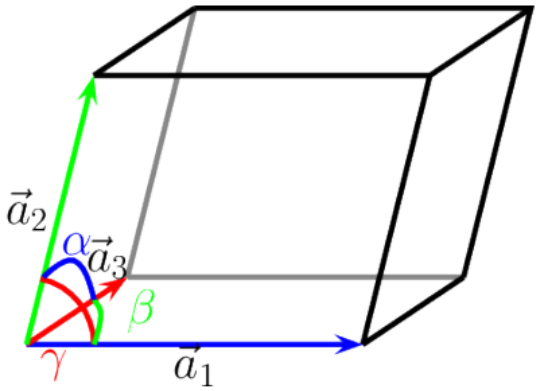
Idealni kristal možemo zamisliti kao prostornu tvorevinu dobivenu beskonačnim ponavljanjem jednakih strukturnih jedinica [11]. Te strukturne jedinice su atomi ili grupe atoma. Važno je naglasiti da se pri definiciji idealnog kristala pretpostavlja da atomi miruju u svojim ravnotežnim položajima.

Idealna kristalna rešetka je konstruirana od grupe atoma koje zovemo *baza*. Temeljno svojstvo takve rešetke je *translacijska invarijantnost*. Baza predstavlja skup tri nekomplanarna vektora $\overrightarrow{a_1}$, $\overrightarrow{a_2}$, $\overrightarrow{a_3}$ sa svojstvom da se u okolini ne mijenja raspored atoma ukoliko se od proizvoljne točke u kristalu pomaknemo za *translacijski vektor rešetke* [11]. Translacijski vektori su definirani u jednadžbi 2.1.

$$\overrightarrow{R} = \sum_{n=1}^{3} n_i \overrightarrow{a_i} \qquad n_i = 0, \pm 1, \pm 2, ... \tag{2.1}$$

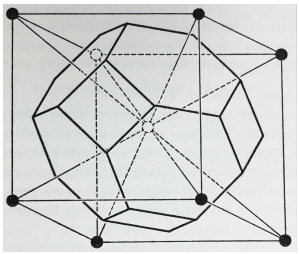
Vektore $\overrightarrow{a_1}$, $\overrightarrow{a_2}$, $\overrightarrow{a_3}$ nazivamo primitivnim translacijskim vektorima ukoliko $\overrightarrow{R}$ povezuje sve ekvivalentne točke kristala. Translacijski vektori određuju smjerove kristalografskih osi. Pri tome oni ne moraju tvoriti ortogonalan sustav.

Paralelopiped koji definiramo preko translacijskih vektora zovemo *primitivna ćelija* [8]. Određujemo je kao nedjeljivu strukturnu volumnu jedinicu kristalne rešetke. Broj atoma u primitivnoj ćeliji je uvijek isti za danu kristalnu strukturu. Bazu kojom je opisana primitivna ćelija zovemo *primitivna baza*. Na slici 2.3 prikazan je paralelopiped koji predstavlja primitivnu (elementarnu) ćeliju. Opisan je uporabom tri kuta i tri stranice.



Slika 2.3: Jedan od mogućih oblika primitivne ćelije [12]

Jednostavna kristalna ćelija se može definirati i tako da polazimo od točke rešetke (čvora) koji je u središtu. Spojnice se povlače iz tog čvora prema najbližim susjednim čvorovima, a pri tome ih okomite ravnine sijeku na njihovom polovištu. Ovakav poliedar se naziva *Wigner-Seitzovova ćelija* [11].



Slika 2.4: Wigner-Seitzova ćelija [13]

2.3 Simetrije kristalne rešetke

U prirodi postoji mnoštvo različitih kristalnih struktura. Gustoća slaganja atoma je različita za različite kristalne strukture. Broj najbližih susjeda koji imaju jednaku udaljenost od nekog elementa rešetke (atoma ili iona) zovemo *koordinacijskim brojem kristalne strukture.*

Kristali su anizotropna stredstva, što znači da fizikalna svojstva mogu imati različite vrijednosti kada se mjerenje vrši u različitim pravcima. Gustoća distribucije čestica u nekom pravcu ovisi o kutovima koji povezuju taj pravac i kristalografske osi. Iz toga proizlazi da se različita fizikalna svojstva, poput termičkih, magnetnih i električnih osobina, mogu mijenjati s promjenom pravca promatranja. Takva anizotropija ne vrijedi za sva svojstva.

Kristalne strukture se mogu preslikati na same sebe preko operatora simetrije. Primjer operacije simetrije je rotacija oko osi koja prolazi kroz točku rešetke [8].

Rotacija, refleksija i inverzija predstavljaju tri tipa operacija simetrije. Rešetke mogu imati jednostruku, dvostruku, trostruku, četverostruku i šesterostruku rotacijsku os. Peterostruke rotacijske osi postoje kod kvazikristala.

- *Jednostruka* (2π) rotacijska os predstavlja onu rotaciju gdje je objektu potrebno 360° da se vrati u početni položaj.
- *Dvostruka* (π) rotacijska os predstavlja onu rotaciju gdje je objekt preslikan sam na sebe nakon rotacije od 180°, što znači da se preslikava dva puta za vrijeme pune rotacije. Primjer je pravokutnik.
- *Trostruka* ($\frac{2\pi}{3}$) rotacijska os predstavlja onu rotaciju gdje je objekt preslikan sam na sebe nakon rotacije od 120°, što znači da se preslikava tri puta za vrijeme pune rotacije. Primjer je jednakostraničan trokut.
- *Četverostruka* ($\frac{\pi}{2}$) rotacijska os predstavlja onu rotaciju gdje je objekt preslikan sam na sebe nakon rotacije od 90°, što znači da se preslikava četiri puta za vrijeme pune rotacije. Primjer je kvadrat.
- *Šestorostruka* ($\frac{\pi}{3}$) rotacijska os predstavlja onu rotaciju gdje je objekt preslikan sam na sebe nakon rotacije od 60°, što znači da se preslikava šest puta za vrijeme pune rotacije. Primjer je šesterokut.

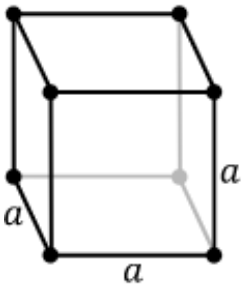
Uz rotacijsku simetriju, važne su translacijska i refleksijska simetrija. Ako idealni kristal transliramo za translacijski vektor $\overrightarrow{R}$, onda se on preslikava sam u sebe. Simetriju zrcaljenja (refleksiju) možemo zamisliti kao rezanje objekta na pola, gdje jednu polovicu priložimo uz zrcalo. Ako na zrcalu vidimo drugu (odsječenu) polovicu objekta, kažemo da je objekt zrcalno simetričan. Ako zrcalo gledamo kao ravninu, tu ravninu zovemo ravninom simetrije kristala.

2.4 Kristalografski sustavi i Bravaisove rešetke

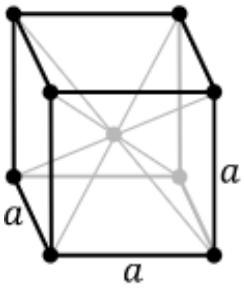
Braviasova rešetka je sustav točaka koji nastaje translacijom za translacijski vektor $\overrightarrow{R}$. Analiza simetrije kristala je pokazala da postoji 14 Bravaisovih rešetki. One su grupirane u sedam kristalografskih sustava: kubični, tetragonski, ortorompski, trigonski, triklinski, monoklinski i heksagonski. Kod triklinskog sustava su svi parametri rešetke različiti, a kubični sustav je najsimetričniji [11]. Tablica 2.1 prikazuje specifičnosti kristalografskih sustava, dok slike 2.4-2.17 prikazuju sve Bravaisove rešetke [14].

Sustav	Kutovi	Stranice
Kubični	$\alpha = \beta = \gamma = 90°$	$a = b = c$
Tetragonski	$\alpha = \beta = \gamma = 90°$	$a = b \neq c$
Ortorompski	$\alpha = \beta = \gamma = 90°$	$a \neq b \neq c$
Trigonski	$\alpha = \beta = \gamma \neq 90°$	$a = b = c$
Heksagonski	$\alpha = \beta = 90°, \gamma = 120°$	$a = b \neq c$
Monoklinski	$\alpha = \gamma = 90° \neq \beta$	$a \neq b \neq c$
Triklinski	$\alpha \neq \beta \neq \gamma \neq 90°$	$a \neq b \neq c$

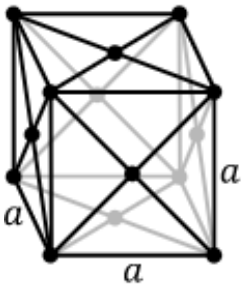
Tablica 2.1: Kristalografski sustavi [11]



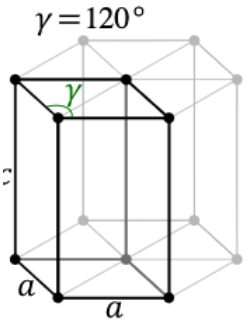
(a) Kubična jednostavna



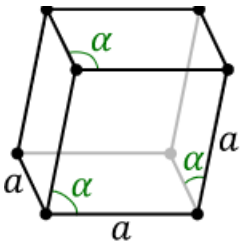
(b) Kubična prostorno centrirana



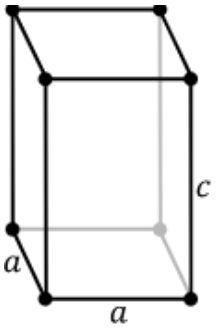
(c) Kubična plošno centrirana



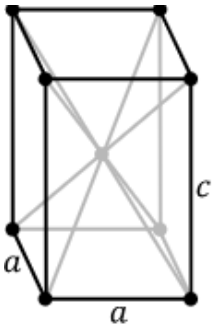
(d) Heksagonska



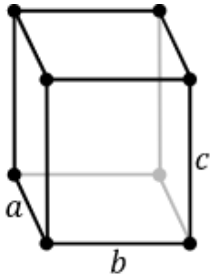
(e) Trigonska



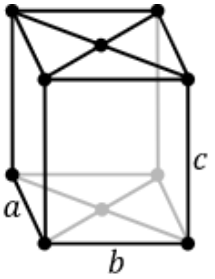
(f) Primitivna tetragonska



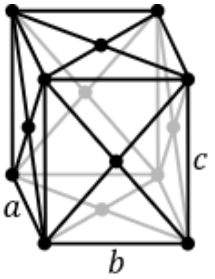
(g) Tetragonska prostorno centrirana



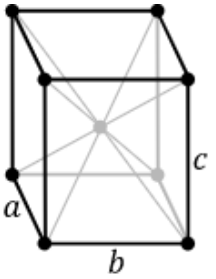
(h) Ortorompska jednostavna



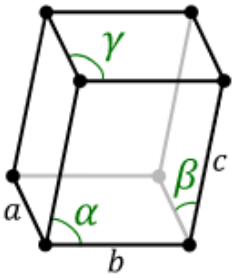
(i) Ortorompska bazno centrirana



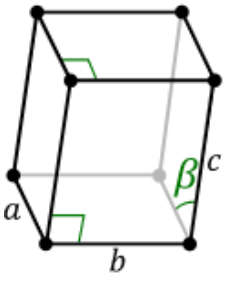
(j) Ortorompska plošno centrirana



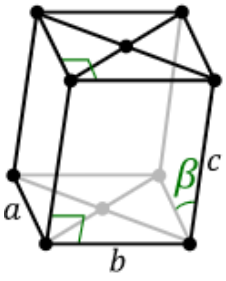
(k) Ortorompska prostorno centrirana



(l) Triklinska jednostavna



(m) Monoklinska jednostavna



(n) Monoklinska bazno centrirana

Slika 2.5: Bravaisove rešetke [14]

2.5 Kubični kristali

Kristali često imaju kubičnu strukturu. Na Slikama 2.5 (a), (b) i (c) vidimo da postoje tri vrste kubične rešetke: jednostavna (engl. simple cubic, SC), prostorno centrirana (engl. body centered cubic, BCC) i plošno centrirana (engl. face centered cubic, FCC). Matematički opisujemo rešetku s primitivnim translacijskim vektorima $\overrightarrow{a_1}$, $\overrightarrow{a_2}$, $\overrightarrow{a_3}$, pri čemu je skalar a konstanta rešetke, a $\hat{x}$, $\hat{y}$, $\hat{z}$ ortonormalni vektori.

2.5.1 Jednostavna kubična rešetka

Svaki od čvorova rešetke dijeli osam kocaka. Definiramo jednostavne translacijske vektore kubične kristalne rešetke.

$$\overrightarrow{a_1} = a \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \overrightarrow{a_1} = a \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \overrightarrow{a_1} = a \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.2}$$

Koristeći primitivne vektore, možemo izračunati volumen jednostavne kristalne ćelije

$$V_c = \overrightarrow{a_1} \cdot (\overrightarrow{a_2} \times \overrightarrow{a_3}) = a^3 \tag{2.3}$$

Sve stranice kubične rešetke su jednake: $a = b = c$. Isto vrijedi za kutove: $\alpha = \beta = \gamma = \frac{\pi}{2}$. U prirodi se rijetko nalaze kristali sa stabilnom jednostavnom kubičnom rešetkom. Poznato je da polonij (Po) u α-fazi ima jednostavnu kubičnu strukturu. Svaki atom u rešetki ima šest najbližih susjeda te je šest koordinacijski broj. Atomi u ovoj rešetki zauzimaju samo oko 52% prostora [15].

2.5.2 Plošno centrirana kubična rešetka

Kod plošno centrirane kubične rešetke, u usporedbu s jednostavnom kubičnom, postoji dodatni čvor na svakoj plohi. Ova struktura je karakteristična za metalne elemente (npr. aluminj (Al), bakar (Cu), srebro (Ag), zlato (Au)). To je visoko stabilna struktura zbog velikog broja susjeda za pojedini čvor. Plošne kubične rešetke su gusto pakirane (engl. close-packed).

Definiramo jednostavne translacijske vektore plošno centrirane kubične kristalne rešetke:

$$\overrightarrow{a_1} = \frac{a}{2} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \qquad \overrightarrow{a_1} = a \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \qquad \overrightarrow{a_1} = a \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \tag{2.4}$$

Volumen primitivne kristalne ćelije je:

$$V_c = \frac{a^3}{4} \tag{2.5}$$

Udaljenost najbližih susjeda jednaka je $\frac{a}{\sqrt{2}}$, dok je koordinacijski broj jednak 12.

2.5.3 Prostorno centrirana kubična rešetka

Kod prostorno centrirane kubične rešetke, u usporedbi s jednostavnom kubičnom, postoji dodatni čvor u centru. Primjeri kristala s ovom strukturom su metali poput željeza (Fe), kroma (Cr) i volframa (W). Koordinacijski broj je 8, udaljenost susjedna dva čvora je jednaka $a\frac{\sqrt{3}}{2}$. Jednostavni translacijski vektori su:

$$\overrightarrow{a_1} = \frac{a}{2} \cdot \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \qquad \overrightarrow{a_1} = \frac{a}{2} \cdot \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \qquad \overrightarrow{a_1} = \frac{a}{2} \cdot \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \tag{2.6}$$

Volumen primitivne kristalne ćelije je:

$$V_c = \frac{a^3}{2}. \tag{2.7}$$

2.6 Heksagonska rešetka

Heksagonska gusto pakirana (engl. hexagonal closest packed, HCP) rešetka sastavljena je od pravilnih šesterostranih prizmi. Pored čvorova u vrhovima, postoje i čvorovi u središtima baza i ploha. Šesterostranu prizmu možemo raščlaniti na tri jednake četverostrane prizme preko kojih definiramo primitivnu kristalnu ćeliju ove strukture [11]. Ćeliju također definiraju kutovi $\alpha = \beta = \frac{\pi}{2}, \gamma = \frac{2\pi}{3}$ i stranice $a \neq b \neq c$. Jednostavni translacijski vektori su:

$$\overrightarrow{a_1} = a \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad \overrightarrow{a_1} = \frac{a}{2} \cdot \begin{pmatrix} -1 \\ \sqrt{3} \\ 0 \end{pmatrix} \qquad \overrightarrow{a_1} = c \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.8}$$

Pri tome je stranica baze a, a visina prizme $c > a$. Koristeći primitivne vektore možemo izračunati volumen jednostavne kristalne ćelije za ovu strukturu:

$$V_c = \overrightarrow{a_1} \cdot (\overrightarrow{a_2} \times \overrightarrow{a_3}) = \frac{a^2}{2} \cdot c \cdot \sqrt{3} \tag{2.9}$$

Možemo zamisliti sitaciju gdje imamo dvije heksagonske podrešetke koje su međusobno pomaknute za vektor:

$$\overrightarrow{d} = \frac{2}{3}\overrightarrow{a_1} + \frac{1}{3}\overrightarrow{a_2} + \frac{1}{2}\overrightarrow{a_3} \tag{2.10}$$

Uvrštavanjem gore definiranih jednostavnih translacijskih vektora dobijemo:

$$\overrightarrow{d} = \frac{1}{2}\begin{pmatrix} a \\ \frac{a}{\sqrt{3}} \\ c \end{pmatrix} \tag{2.11}$$

Daljnim sređivanjem dobijemo:

$$\overrightarrow{d} = \sqrt{\frac{a^2}{3} + \frac{c^2}{4}} \tag{2.12}$$

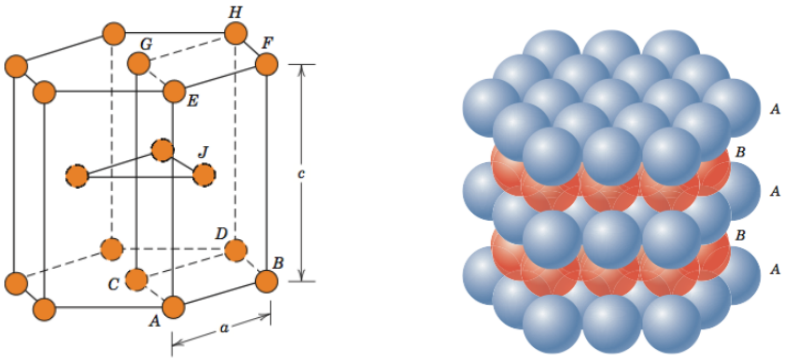
Uzevši da je pomak $\overrightarrow{d}$ jednak stranici baze a dobijemo omjer stranice baze i visine prizme [11]:

$$\frac{c}{a} = \sqrt{\frac{8}{3}} \tag{2.13}$$

Primjeri HCP struktura su kobalt (Co), kadmij (Cd), cink (Zn) i α faza titanija (Ti) [16].

2.7 Gusto pakirane rešetke

Pri razmatranju strukture kristala možemo zamisliti atome kao jednostavne, čvrste sfere. Najučinkovitiji način za smještanje tih sfera je da je prazan prostor izmedju njih minimiziran. Ovakav tip pakiranja se zove gusto pakiranje (engl. close-packing). Ako pogledamo prvi sloj sfera na Slici 2.6, vidimo da svaka sfera dodiruje 6 susjednih. Ako na prethodni sloj dodamo sljedeći, taj sloj se također gusto pakira. U svrhu minimiziranja praznog prostora izmedju slojeva, drugi sloj sjedi na udubljenima izmedju sfera prvog sloja. Atomi trećeg sloja mogu biti postavljeni iznad udubina u prvom sloju što zovemo *ABC* uređenje (svako slovo definira poziciju sloja). Medjutim, atomi trećeg sloja mogu biti postavljeni i direkno iznad atoma prvog sloja, sto zovemo *ABA* uređenje. Kada se ovakva udređenja ponavljaju, oni nam definiraju dva tipa gusto upakiranih struktura. ABCABC pakiranje zovemo *gusto kubično pakiranje,* dok ABABAB zovemo *gusto heksagonsko pakiranje*. Eksperimentalno je pokazano da FCC i HCP rešetke jednako gusto smještaju atome koji okupiraju 74% prostora.



(a) (b)

Slika 2.6: Gusto pakirana heksagonska struktura [16]

3 Strojno učenje

3.1 O strojnom učenju

Tom M. Mitchell je u svojoj poznatoj knjizi napisao ovu definiciju strojnog učenja: *"Kaže se da računalni program uči iz iskustva E s obzirom na neku klasu zadataka T i mjera izvedbe P, ako se izvedba na zadacima u T, mjereno pomoću P, poboljšava s iskustvom E"* [1]. Ova definicija uključuje očekivanja i provjere valjanosti. Motivacija znanstvenika je bio razvoj modela u kome računalo samo rješava probleme. Ti modeli su u prvom periodu razvoja strojnog učenja najčešće bili perceptroni. Taj naziv je 1957. godine izabrao Frank Rosenblatt [17]. Perceptron je osnova razvoja današnjih neuronskih mreža.

Veliki iskorak je napravljen krajem prošlog stoljeća kada je strojno učenje postalo samostalna disciplina. Fokus se maknuo od simboličkog pristupa prema matematičkim modelima iz vjerojatnosti i statistike. No matematički aparat je tu samo dio priče. Ogroman doprinos za razvoj algoritama čini dostupnost velike količine podataka i računalnih programa.

Glavne metode strojnog učenja su: *nadzirano (engl. supervised), nenadzirano (engl. unsupervised) i podržano (engl. reinforcement)* [18–20]. Kod nadziranog učenja za program se priprema skup ulaznih podataka i skup očekivanih vrijednosti. Cilj je naći korelaciju izmedju ulaznih i izlaznih podataka [20]. Učenje bez nadzora karaterizira nepostojanje izlaznog skupa podataka, već algoritam sam mora odrediti strukturu ulaznih podataka. Kod podržanog učenja program stupa u interakciju s dinamičkim okruženjem u kojem mora postići određeni cilj [19].

U ovome radu se koriste dvije metode nadziranog strojnog učenja: linearna regresija i neuronske mreže.

3.2 Nadzirano strojno učenje

U nadziranom strojnom učenju podatci koji opisuju zadani problem se podijele na skup za treniranje (engl. training set) i skup za testiranje (engl. test set). Skup za treniranje se koristi za konstrukciju modela i sadrži izlazne podatke za sve ulazne podatke. U praksi je za programera često vrlo zahtjevno formirati takvu listu ulaznih i izlaznih podataka. To je vrlo važan korak zbog toga što model uči na parovima ulazno-izlaznih podatka. Skup za testiranje se koristi za ispitivanje kvalitete modela. Poznat primjer nadziranog strojnog učenja je kategorizacija slika. U takvom postupku možemo naučiti računalni program da razlikuje fotografije objekata ili ljudskih lica.

Možemo koristiti notaciju $x^{(i)}$ za ulazne varijable (engl. features), te $y^{(i)}$ za izlazne varijable (engl. target). Skup ($x^{(i)}$, $y^{(i)}$) je skup za treniranje [18, 19]. Matematički

gledano, svaki testni primjerak je prikazan kao vektor, dok je skup za treniranje prikazan kao matrica. Algoritmi nadziranog strojnog učenja preko iterativne optimizacije ”uče” funkciju da može predvidjeti skup izlaznih podataka na temelju skupa ulaznih podataka [18, 19]. Optimalno trenirana funkcija će omogućiti algoritmu da predvidi izlazne podatke koji nisu pripadali skupu za treniranje. Ukoliko je varijabla koju želimo predvidjeti kontinuirana, takav problem nazivamo *regresijskim*. Kada takva varijabla poprima diskretne vrijednosti, imamo *klasifikacijski problem*. Naš cilj je da na skupu za treniranje nadjemo funkciju $h : X \to Y$, tako da ta funkcija h daje dobru generalizaciju problema koja će naći korelaciju izmedju ulaznih i izlaznih podataka. Funkciju h zovemo *hipotezom*.

3.3 Linearna regresija

Linearna regresija je statistička metoda koja se često koristi za predviđanja i provjere modela [18,19,21,22]. Jednostavan primjer je procjena kupnje stana u nekom gradu. Primjer podataka je prikazan u Tablici 3.1.

Broj soba	Površina stana (m^2)	Cijena (EUR/m^2)
2	52	1900
4	110	2800
2	48	2000
3	95	2350
...	...	...

Tablica 3.1: Primjer skupa podataka koji se mogu koristiti za linearnu regresiju

Na osnovu ovakvih podataka (engl. dataset) želimo predvidjeti cijenu u ovisnosti o broju soba ili površini stana. Cijena stana može biti bilo koji broj te je ovaj problem regresijski. Pri rješavanju problema odlučujemo što su ulazne varijable, koje u strojnom učenju zovemo značajke (engl. features). U ovom primjeru neka je $x_1^{(i)}$ broj soba i-tog stana, dok je $x_2^{(i)}$ njegova površina. Vrijednosti $(x_1^{(i)}, y^{(i)})$ ili $(x_2^{(i)}, y^{(i)})$ predstavljaju skup za treniranje. Možemo prikazati nepoznatu vrijednost stana y kao linearnu funkciju x i zbog toga je ovaj model linearna regresija. Želimo odrediti *hipotezu* h koja omogućava dobro predviđanje cijene stana:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2, \tag{3.1}$$

pri čemu θ_i predstavljaju težinske parametre koji parametriziraju linearno preslikavanje iz domene u kodomenu.

Slično razmatranje vrijedi za primjer iz fizike [21]. Možemo promotriti loptu bačenu vertikalno uvis i uzeti izmjerene visine h_i u vremenu t_i. Ukoliko zanemarimo otpor zraka, ovaj problem možemo prikazati linearnom regresijskom funkcijom

koja je prikazana u 3.2. To znači da korištenjem izmjerenih veličina (h_i,t_i), možemo procijeniti vrijednosti početnih brzina i gravitacijske konstante.

$$h_i(t) = \theta_1 t_i + \theta_2 t_i^2 + \epsilon_i \tag{3.2}$$

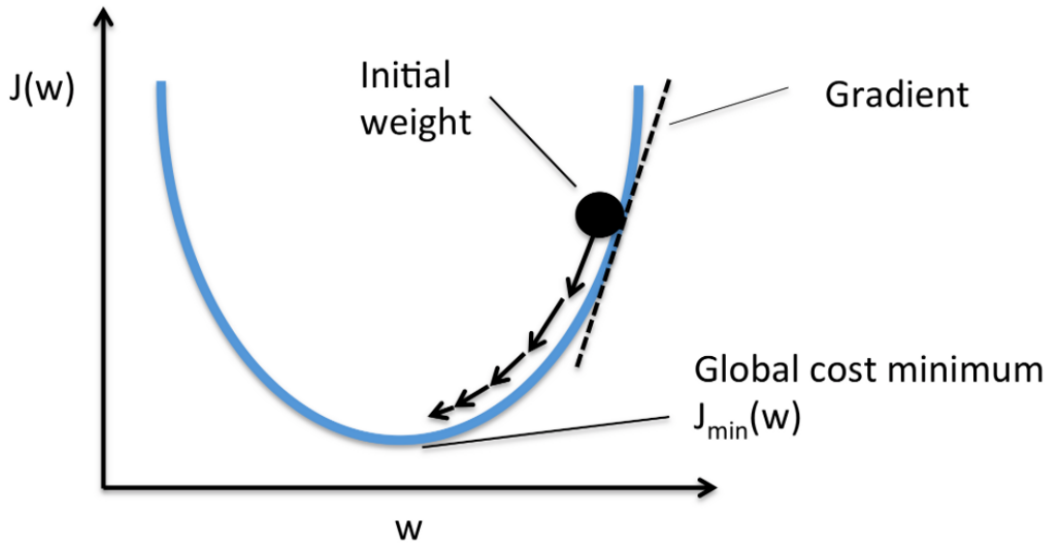
U jednadžbi 3.2 pocetna brzina je θ_1, θ_2 je proporcionalna gravitacijskoj konstanti, a ϵ_i opisuje greške mjerenja.

Postoje četiri pretpostavke za regresijski model [23]. To su: linearnost, konstantna varijanca, neovisnost i normalnost. Linearnost znači da je odnos elemenata domene X i srednje vrijednosti Y linearan. Pod konstantnom varijancom podrazumijevamo da je za sve vrijednosti X ista varijanca u njihovim pogreškama. Pored toga izlazne vrijednosti su neovisne jedna od druge. Za bilo koju fiksnu vrijednost X, vrijednosti Y su normalno distribuirane.

3.3.1 Minimizacija funkcije troška

Želimo opisati postupak učenja, tj. određivanja parametara θ_i. U 3.3 definirana je funkcija troška (engl. cost-function) [22]. Prikazana je shematski na Slici 3.1.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{3.3}$$



Slika 3.1: Funkcija troška [24]

Ako je vrijednost funkcije troška velika, to znači da su rezultati daleko od očekivanih. Želimo da nam vrijednosti $h_\theta(x)$ budu što bliže izlaznim vrijednostima y. Funkcija troška mjeri koliko dobro model reprezentira odnos ulaznih i izlaznih podataka [25]. Možemo reći da funkcija troška predstavlja udaljenost između stvarne i predviđene

vrijednosti. Cilj algoritama strojnog učenja je naći parametre ili, generalnije rečeno, strukturu koja minimizira funkciju troška.

Linearna regresija nam ne govori kako se određuju parametri, već samo pretpostavlja linearnu ovisnost zavisne i nezavisne varijable. Želimo trenirati parametre θ da bi smo dobili što manju vrijednost $J(\theta)$. U praksi prvo proizvoljno zadajemo početnu vrijednost θ, te *gradijentnim spustom* (engl. gradient descent) postižemo konvergenciju ka željenoj vrijednosti. Gradijentni spust je vrsta optimizacijskog algoritma koji ima cilj naći globalni minimum konveksne funkcije. Matematički opis gradijentnog spusta je predstavljen izrazom 3.4, gdje znak ":=" predstavlja konstantno ažuriranje, d broj ulaznih varijabli, a α stopu učenja (engl. learning rate):

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad j \in \{0, 1, ..., d\} \tag{3.4}$$

Ideja ovog algoritma je jednostavna: konstantno se traži korak koji vodi ka smjeru najstrmijeg pada. Moramo izračunati parcijalnu derivaciju. Krenut ćemo od derivacije jednog primjerka iz skupa za treniranje (x,y) [22]:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d} \theta_i x_i - y \right) \\ &= (h_\theta(x) - y) x_j \end{aligned} \tag{3.5}$$

Izveli smo parcijalnu derivaciju za jedan primjerak iz skupa za treniranje. Iz te formule možemo dobiti ažuriranu vrijednost za jedan primjerak iz skupa za treniranje:

$$\theta_j := \theta_j + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)} \tag{3.6}$$

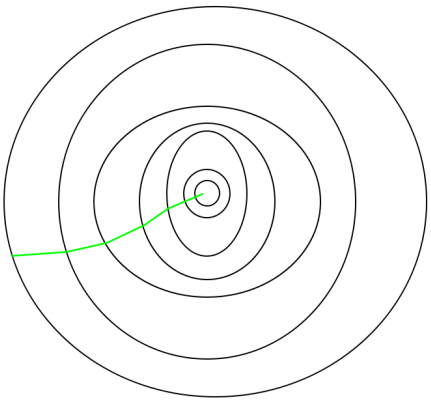
U jednadžbi 3.6 član $(y^{(i)} - h_\theta(x^{(i)}))$ predstavlja pogrešku. "Korak" ažuriranja je proporcionalan pogrešci. Ukoliko se rezultat (predviđanje) približno podudara s traženom vrijednošću (engl. target), parametri se ne moraju puno mijenjati. Isto vrijedi i suprotno, ukoliko imamo veliku grešku, parametri se moraju značajno mijenjati. Ovim smo objasnili kako funkcionira algoritam za jedan primjerak iz skupa za treniranje (x,y).

Za sve primjerke j iz skupa za treniranje možemo iterirati funkciju do njene konvergencije. Takav algoritam se zove batch gradijentni spust (engl. batch gradient descent, BGD). Riječ *batch* se odnosi na grupu primjeraka za treniranje koju koris-

timo u jednoj iteraciji. Umjesto pisanja svih primjeraka θ_j možemo koristiti vektorsku notaciju θ i napisati formulu za batch gradijentni spust [22]:

$$\theta := \theta + \alpha \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)}))x^{(i)} \tag{3.7}$$

Slika 3.2 prikazuje moguću putanju batch gradijentnog spusta, gdje su zelenom linijom spojene uzastopne vrijednosti θ, a elipse su konture kvadratne funkcije.



Slika 3.2: Putanja "batch" gradijentnog spusta [26]

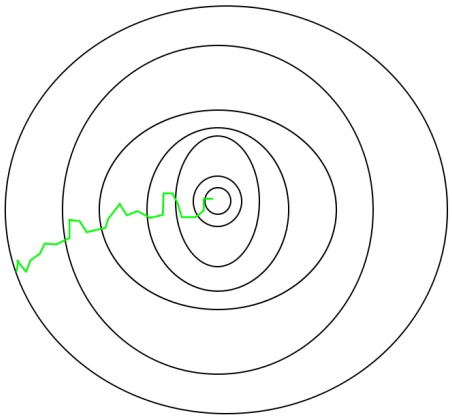
Druga metoda je stohastički gradijentni spust (engl. stohastic gradient descent). Za razliku od batch gradijentnog spusta koji mora proći kroz čitav skup za treniranje da bi krenuo na idući korak (što ima visok računalni trošak ukoliko je skup veliki), stohastička verzija algoritma više puta prolazi kroz skup za treniranje. U trenutku kada obrađuje određeni primjerak, parametri θ se ažuriraju u skladu s greškom $(y^{(i)} - h_\theta(x^{(i)}))$ samo za taj primjerak. Stohastičkim algoritmom se brže stiže do minimuma, no s druge strane postoji mogućnost da ta metoda nikad ne nađe minimum. U tom slučaju dolazi do oscilacija oko minimuma. U praksi se češće koristi stohastički algoritam. Taj algoritam treba koristiti samo kada je ažuriranje provedeno nakon svake evaluirane grupe iz promatranog skupa podataka [17]. Stohastički gradijentni spust se može predstaviti formulom [22]:

$$\theta := \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}. \tag{3.8}$$

Slika 3.3 prikazuje putanju stohastičkog gradijentnog spusta.

3.3.2 Linearna regresija u programskom jeziku Python

Jednadžba 3.6 poznata je u matematičkoj literaturi kao pravilo najmanje srednje kvadratne vrijednosti (engl. least mean squares rule, LMS) ili Widrow-Hoff pravilo

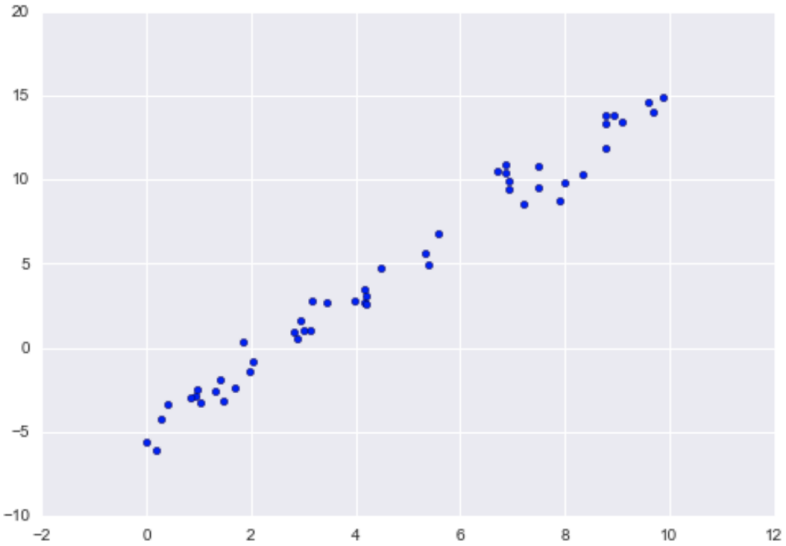


Slika 3.3: Putanja stohastičkog gradijentnog spusta [26]

učenja [22]. Međutim, linearna regresija se kao metoda najmanjih kvadrata često primjenjuje u znanstvenim i tehničkim disciplinama za prilagodbe funkcija podatcima (tzv. "fitanje"). Scipy, osnovna bibliteka za numerički rad u Pythonu, sadrži neke programe za prilagodbu funkcija podatcima [27]. U scikit-learn, programskom paketu za strojno učenje u Pythonu, nalaze se programi za regresiju [28].

Jednostavnu linearnu regresiju možemo opisati jednadžbom $y = ax + b$, pri čemu je a nagib, dok b predstavlja odsječak na y-osi. Za potrebe primjera uzet ćemo podatke koji su slučajno raspršeni oko linije definirane koeficijentima $a = 2$ i $b = -5$ [29]. Podaci i programski kod koji ih generira su prikazani na Slici 3.4.

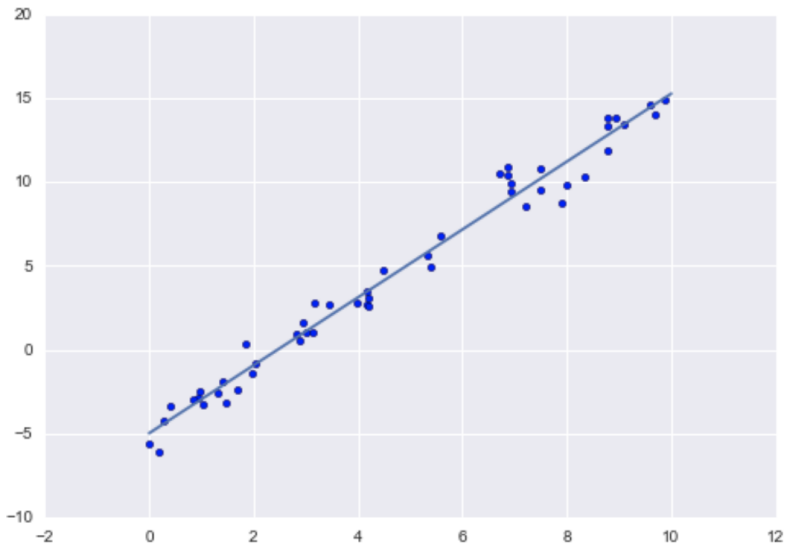
```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)
plt.scatter(x, y);
```



Slika 3.4: Prikaz podataka [29]

Korištenjem programa *LinearRegression*, koji se nalazi u sckit-learn paketu, možemo nacrtati liniju koja najbolje opisuje zadane podatke. Progamski kod i rezultat su prikazani na Slici 3.5.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)

model.fit(x[:, np.newaxis], y)

xfit = np.linspace(0, 10, 1000)
yfit = model.predict(xfit[:, np.newaxis])

plt.scatter(x, y)
plt.plot(xfit, yfit);
```



Slika 3.5: Konstrukcija regresijske linije [29]

Uporabom *LinearRegression* mogu se analizirati i višedimenzionalni linearni modeli oblika $y = a_0 + a_1x_1 + a_2x_2 + \ldots$. U takvim primjerima određuje se hiper-ravnina u prostorima više dimenzija. Često se koriste i modeli nelinearne regresije.

Procjena izvedbe modela Nakon određivanja parametara modela, važno je procijeniti greške. Za regresijske modele najčešće koristimo tri evaluacijske metrike [30]. Sve tri su kao funkcije prisutne u programskom paketu scikit-learn.

1. Srednja apsolutna pogreška (engl. Mean absolute error) je srednja vrijednost apsolutne vrijednosti pogrešaka:

$$\frac{1}{n}\sum_{n=1}^{n} |Stvarna\ vrijednost - Predviđena\ vrijednost| \tag{3.9}$$

2. Srednja kvadratna pogreška (engl. Mean squared error) je srednja vrijednost kvadriranih grešaka:

$$\frac{1}{n}\sum_{n=1}^{n}|Stvarna\ vrijednost - Predviđena\ vrijednost|^2 \tag{3.10}$$

3. Korijenska srednja kvadratna pogreška (engl. Root mean squared error) je korijen srednje vrijednosti kvadriranih grešaka:

$$\sqrt{\frac{1}{n}\sum_{n=1}^{n}|Stvarna\ vrijednost - Predviđena\ vrijednost|^2} \tag{3.11}$$

Podnaučenost i prenaučenost U procesu izrade modela strojnog učenja mogu nastati specifični problemi koje zovemo podnaučenost (engl. underfitting) i prenaučenost (engl. overfitting) [31].

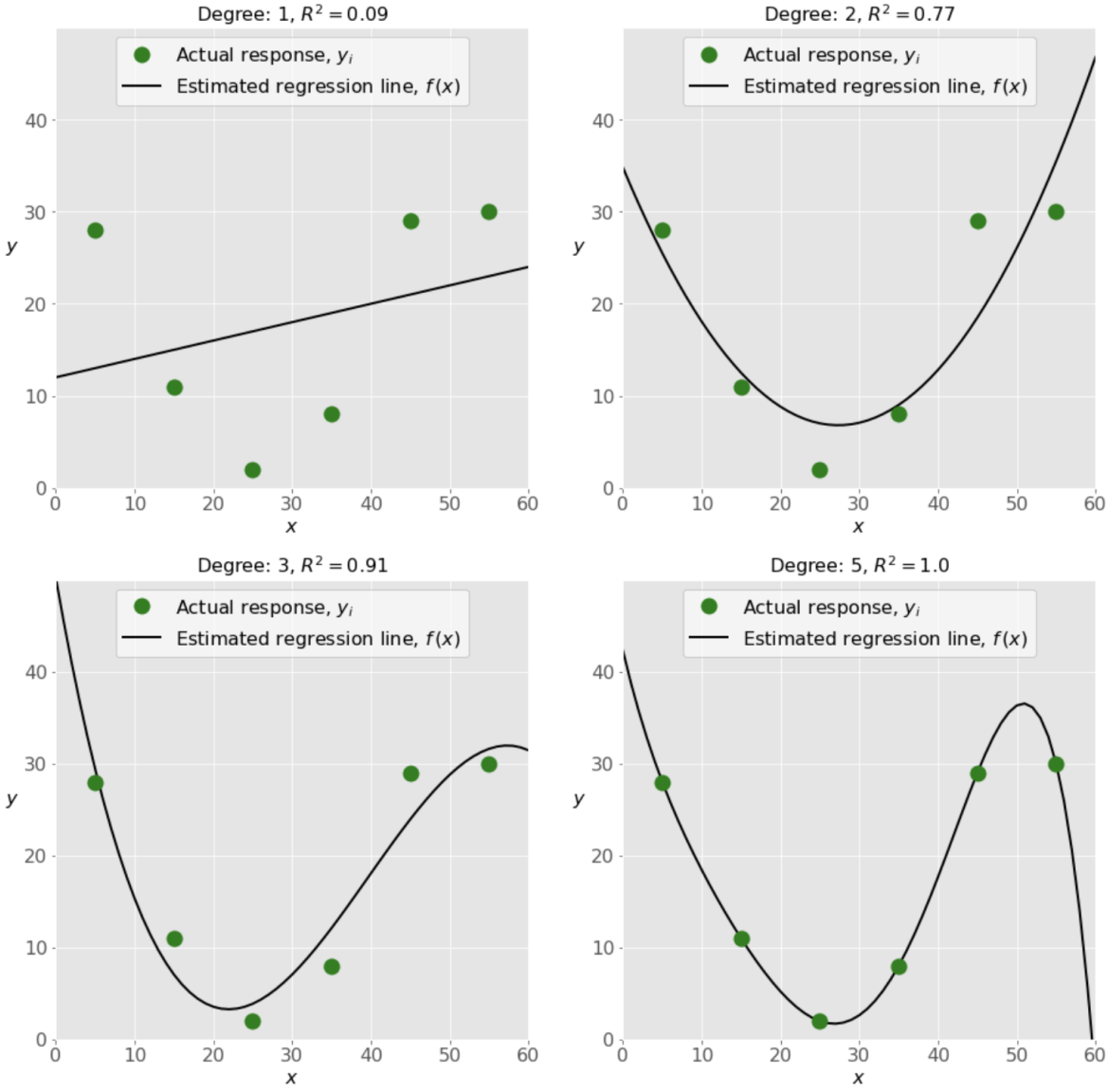
Podnaučenost se događa kada model ne može precizno odrediti međuovisnost podataka, što je najčešće posljedica jednostavnosti tog modela. Može se definirati koeficijent R^2 kao dio promjene zavisne varijable y koji se može objasniti kao ovisnost o neovisnoj varijabli x u tom modelu. Podnaučeni modeli daju nisku vrijednost R^2 i slabu sposobnost generalizacije kada se primjenjuju na novim podatcima [31].

Prenaučenost se događa kada model nauči ovisnosti među podacima zajedno sa slučajnim fluktuacijama. Složeni modeli koji imaju mnoštvo značajki često pokazuju prenaučenost. Kad se primjenjuju na poznate podatke, takvi modeli daju visoke vrijednosti R^2. Međutim, oni često ne generaliziraju dobro i imaju znatno niži R^2 kada se koriste s novim podacima [31].

Slike 3.6 prikazuju regresijske linije za nekoliko vrijednosti koeficijenta R^2 [31]. Na prvoj slici regresijska linija ne opisuje dobro ulazne podatke (zelene točke) i u ovakvom primjeru možemo očekivati podnaučenost. Druga slika je primjer odabira optimalnog regresijskog stupnja. Treća slika prikazuje znakove prenaučenosti. Na toj slici vidimo da kada varijabla x poprimi vrijednosti oko 60, linija počinje padati iako podatci ne upućuju na to. Četvrta slika pokazuje savršeno podudaranje modela i podataka. U malom broju modela takvo idealno podudaranje može biti traženo rješenje, ali najčešće tako nastaje prenaučenost.

3.4 Duboko strojno učenje

Duboko učenje je vrsta reprezentacijskog učenja [32]. Metode takvog strojnog učenja automatski formiraju korisnu reprezentaciju podataka. Značajne su za klasifikacijske i regresijske probleme u različitim područjima [17]. Važan dio dubokog strojnog



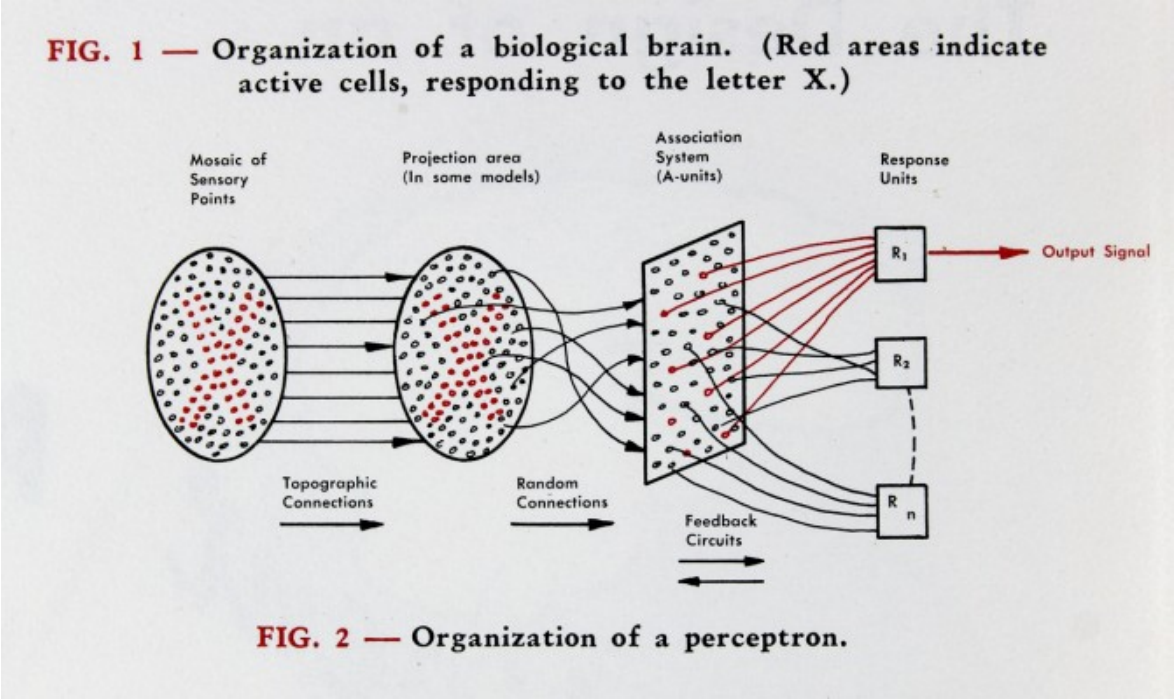
Slika 3.6: Primjeri podnaučenosti, dobre naučenosti i prenaučenosti [31]

učenja su *neuronske mreže*. Modeli neuronskih mreža daju mogućnost brze evaluacije. Međutim, njihova funkcija troška nije konveksna. Zbog toga faza treniranja neuronskih mreža može biti računalno zahtjevna.

3.4.1 Kratka povijest neuronskih mreža

Psiholog Frank Rosenblatt je predložio perceptron kao prvi model neuronskih mreža [17, 33]. Njegova ideja je bila da se mogu napisati računalni algoritmi koji rade slično kao neuroni u mozgu. Rosenblattov preceptron je prikazan na Slici 3.7. Rad perceptrona je 1958. godine demonstriran na IBM 704 računalu. Računalo, vrlo napredno u to vrijeme, bilo je teško pet tona i zauzimalo cijelu sobu. Informacije su se unosile preko bušenih kartica. Bušene kartice predstavljaju komad čvrstog papira i prenose informacije postojanjem ili nepostojanjem rupica na zadanim mjestima [34].

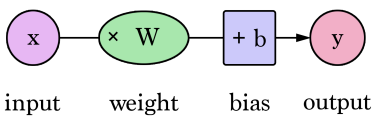
Nakon 50 pokušaja, računalo je naučilo kako razlikovati kartice označene s lijeva od onih koje su označene s desne strane [33]. Danas kažemo da je Rosenblattov perceptron jednoslojna neuronska mreža i algoritam koji klasificira ulazne podatke u dvije kategorije.



Slika 3.7: Skica Rosenblattovog perceptrona [33]

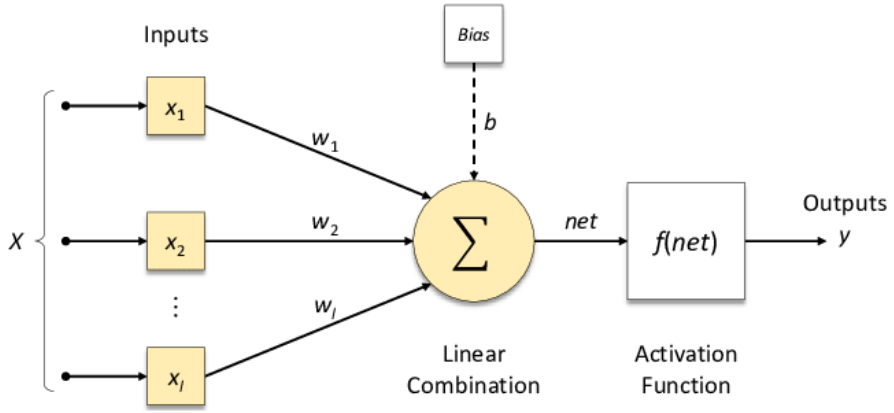
3.4.2 Struktura neuronskih mreža

Najjednostavnija neuronska mreža je prikazana na Slici 3.8 [35]. Pored ulaza i izlaza, ta mreža ima parametre w i b čija se vrijednost određuje u procesu učenja. Pri tome je w težinski parametar (engl. weight), a b je pristranost (engl. bias). Parametri w i b transformiraju ulazne podatke unutar neuronske mreže. Na početku treniranja mreža će nasumično postaviti vrijednosti parametara w i b. Tijekom rada ti parametri se mijenjaju, pri čemu se rezultat usmjerava prema željenoj vrijednosti. Niske vrijednosti parametara w i b imaju mali utjecaj na promjenu ulaza, dok će visoke vrijednost imati značajniji utjecaj na izlaz.

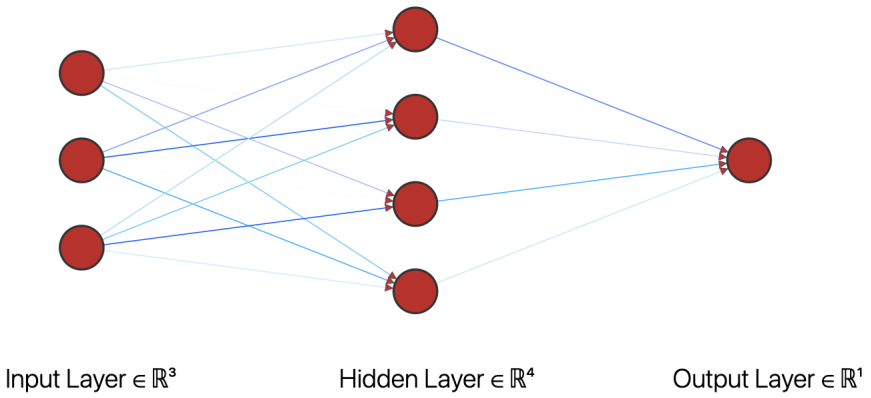


Slika 3.8: Shema jednostavne neuronske mreže [35]

Slika 3.9 prikazuje strukturu složenije neuronske mreže. Definiramo ulazni skup podataka $X \in \{x_1, x_2, ...x_n\}$ koji zovemo *ulaznim slojem* (engl. input layer). Neuron označen kao Σ predstavlja *skriveni sloj* (engl. hidden layer). Na kraju je izlaz Y koji predstavlja *izlazni sloj* (engl. output layer). Primjer mreže s više neurona je prikazan na Slici 3.10. Ulazni skup podataka za tu mrežu je $X \in \{x_1, x_2, x_3\}$, Skriveni sloj ima četiri, a izlazni sloj jedan neuron.



Slika 3.9: Osnovna neuronska mreža [36]



Slika 3.10: Primjer složenije neuronske mreže [37]

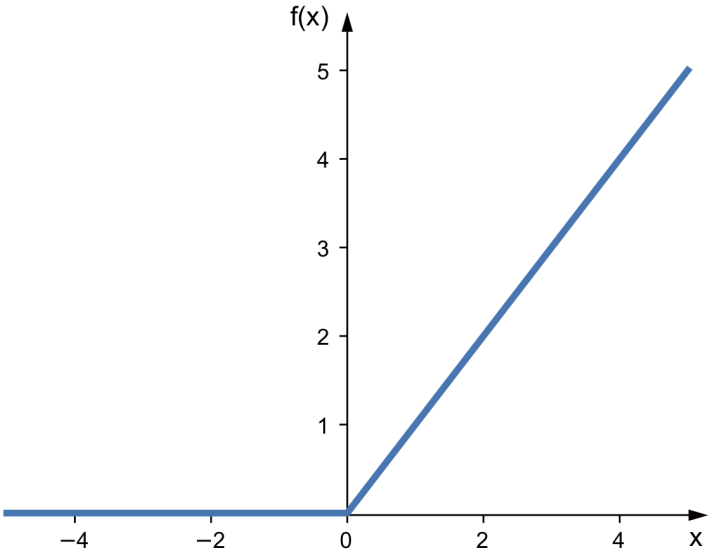
3.4.3 Aktivacijske funkcije

Na Slici 3.9 je prikazana i aktivacijska funkcija. Takve funkcije su u praksi nelinearne i izvršavaju račun u neuronskoj mreži tako što djeluju na linearne kombinacije težinskih parametara i pristranosti [38]. Jedna od ključnih zadaća pri pisanju algoritma i računalnog programa je selekcija optimalne aktivacijske funkcije koja daje dobre rezultate u različitim područjima, poput klasifikacije objekata, prijevoda teksta, ili autonomnih vozila.

Jednostavna aktivacijska funkcija je *ReLU* (engl. rectified linear unit) koja zamjenjuje negativne vrijednosti nulom (vidi jednadžbu 3.12). ReLU aktivacijska funkcija

je prikazana na Slici 3.11.

$$f(x) = max(0, x) \tag{3.12}$$



Slika 3.11: ReLU aktivacijska funkcija

Drugi primjer je hiperbolički tangens koji ulazne vrijednosti preslikava na interval $[-1, 1]$.

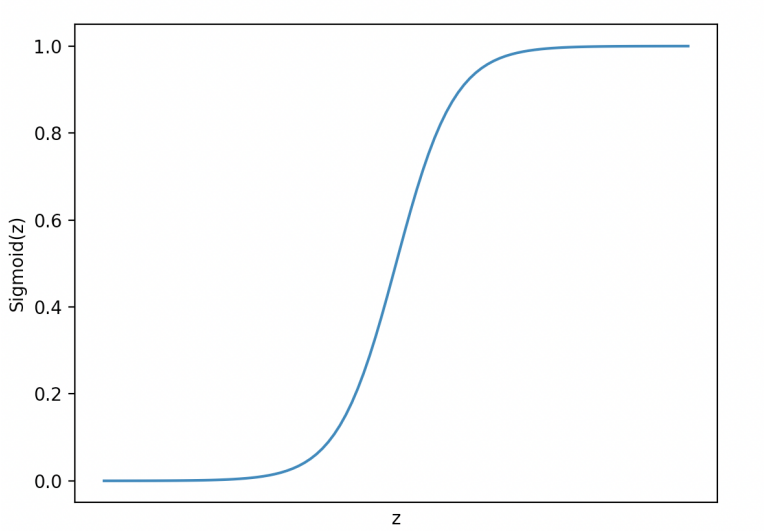
Primjer aktivacijske funkcije je i sigmoidna funkcija koja je definirana u jednažbi 3.13 i prikazana na Slici 3.12 [39]. Ta funkcija preslikava ulazne vrijednosti na interval između nule i jedinice.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.13}$$

Specijalna vrsta aktivacijske funkcije je softmax [40]. To je normirana eksponencijalna funkcija predstavljena jednadžbom 3.14.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}, \quad i = 1, ..., N \tag{3.14}$$

Softmax proizvodi normirani output koji predstavlja vjerojatnost za moguće rezultate neuronske mreže. To znači da softmax izračuna decimalne brojeve čiji je zbroj jednak 1.0. Najčešće se koristi u konačnom skrivenom sloju neuronskih mreža koje rade klasifikaciju. Zanimljivo je da se ova funkcija koristi i u statističkoj fizici kao Boltzmannova raspodjela.



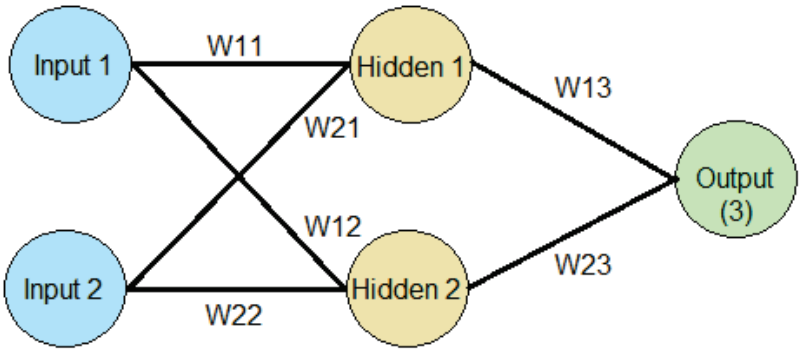
Slika 3.12: Sigmoidna funkcija

3.4.4 Algoritam s povratnim postupkom

Algoritam s povratnim postupkom (engl. backpropagation algorithm) je metoda kojom neuronske mreže uče. Taj algoritam su 1985. godine predložili Rumelhart, Hinton i Williams [41]. U algoritmu s povratnim postupkom neuronska mreža uči iz svojih grešaka. Pri prvom prolazu kroz mrežu računa se predviđanje rezultata za sve primjerke iz skupa za treniranje. Taj postupak se zove prolaz unaprijed (engl. forward pass). Izračuna se odstupanje od očekivanog rezultata. U drugom koraku se prolazi kroz mrežu unatrag (engl. back pass). Pri tome se određuje koliko svaki od neurona doprinosi greški. Vrijednosti težinskih parametara w i pristranosti b se mijenjaju uporabom algoritama gradijentnog spusta.

Primjer povezanosti neurona i težinskih parametara koji se mijenjaju povratnim postupkom prikazan je na Slici 3.13. Može se analizirati model takve neuronske mreže koji ispituje ovisnost neke fizičke veličine (neuron Output) o dvije veličine (neuroni Input 1 i Input 2). Npr. izlazni neuron može biti Youngov modul elastičnosti, a ulazni neuroni temperatura taljenja i električni otpor.

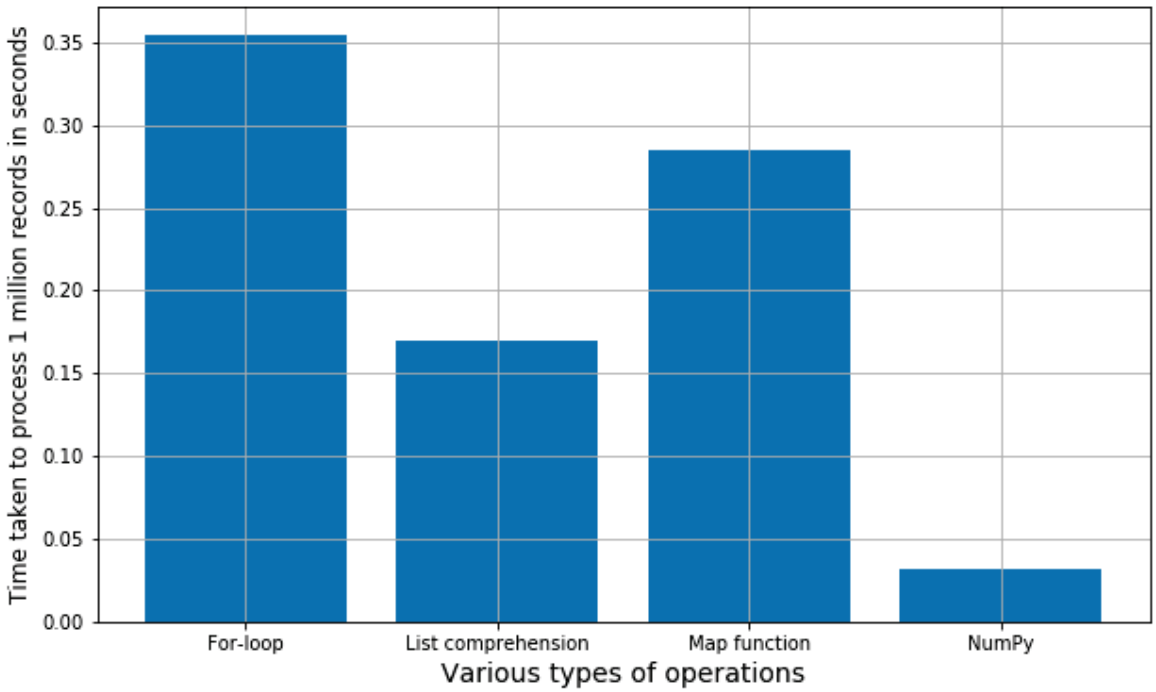
Kao drugi primjer možemo razmotriti treniranje na skupu koji se sastoji od puno fotografija zlatnog nakita. Želimo trenirati neuronsku mrežu tako da uspješno otkriva fotografije prstena, za razliku od npr. narukvice ili lanca. Slike su u računalima reprezentirane numerički, tj. preko niza brojeva. Prstenje iz našeg primjera je predstavljeno odredjenim nizom brojeva koji se razlikuje od niza brojeva za narukvice i lance. Ako je neuronska mreža na početku treniranja, rezultati koje ona daje nisu zadovoljavajući, tj. greške u klasifikaciji zlatnog nakita su velike. Račun se ponavlja, parametri w i b se mijenjaju sve dok greške ne budu manje od unaprijed određene vrijednosti. Npr. u nekom trenutku treniranja rezultat može biti da je za neku fotografiju



Slika 3.13: Primjer neuronske mreže s težinskim parametrima koji se mijenjaju algoritmom s povratnim postupkom [42]

prstena mreža odredila da je to prsten uz vjerojatnost od 87%, 10% je narukvica i 3% je lanac.

U praktičnom radu s neuronskim mrežama koristimo matrične jednadžbe. Takav pristup problemu nazivamo *vektorizacijom*. Tim postupkom uklanjanjamo eksplicitne *for petlje* u programskom kodu. Korištenjem vektorizacije možemo znatno smanjiti vrijeme izvršavanja računalnih programa. Programski paketi, kao što su NumPy [43], TensorFlow [44], ili PyTorch [45], postižu visoku efikasnost operacija nad matricama, a to znantno olakšava treniranje velikih skupova podataka. Na Slici 3.14 je prikazano vrijeme izvršavanja programa s matricama u Pythonu kad se koriste for petlje, dvije funkcije koje su dio funkcionalnog programiranja (list comprehension i map) te programski paket NumPy.



Slika 3.14: Usporedba brzina izvršavanja matematičkih operacija s matricama [46]

4 Istraživanje kristala metala linearnom regresijom i neuronskim mrežama

4.1 Python i programski paketi

4.1.1 Python

Python je dinamički programski jezik koji podržava objektno orijentirano i funkcionalno programiranje [47]. Guido van Rossum je počeo pisati Python krajem 1989. godine kao skriptni jezik za operativni sustav Amoeba. Jezik je dobio ime po poznatom serijalu *"Monty Python"*. Van Rossum je objavio Python 1991. godine.

Python je elegantan, praktičan i jednostavan programski jezik. Najveća mu je snaga u kontinuiranom "open-source" razvoju, gdje programeri diljem svijeta dodaju nove funkcionalnosti. Pyhon se često koristi kao skriptni jezik i za razvoj specijalnih web stranica. Puno se koristi u strojnom učenju i općenito u analizi znanstvenih rezultata. Zajedno s bibliotekama čini vrlo efikasan, praktičan i moćan aparat koji olakšava rad.

4.1.2 NanoHub

NanoHub je mrežni portal zajednice istraživača, nastavnika, učenika i studenata koje zanima nanotehnologija [7]. Ta platforma sadrži lekcije i baze s informacijama o materijalima i nanostrukturama. Veliki broj računalnih modula na NanoHubu daje studentima mogućnosti učenja i istraživanja. U ovom diplomskom radu korišten je modul za strojno učenje koji je još u razvoju [48].

4.1.3 Jupyterove bilježnice

Jupyterove bilježnice su aplikacije koje omogućuju stvaranje i dijeljenje dokumenata koji sadrže programski kod, tekst, jednadžbe i slike [49]. Jednostavne su za upotrebu te se koriste za razvoj programskog koda, numeričke simulacije, statističko modeliranje, strojno učenje i vizualizaciju podataka.

4.1.4 Pandas

Pandas je programski paket za analizu podataka [50]. Prve primjene su bile u financijskim analizama, ali se Pandas vrlo brzo počeo koristiti u programiranju i analizi velike količine podataka. Ime dolazi od *Pan*el *Da*ta, što je termin koji opisuje multidimenzionalne skupove podataka.

4.1.5 TensorFlow

TensorFlow je cjelovita platforma otvorenog koda za strojno učenje. Sadrži sveobuhvatan, fleksibilan ekosustav alata, programskih paketa i resursa koji omogućava i olakšava cjelokupni istraživački proces. Programeri lako grade i implementiraju programe za strojno učenje. TensorFlow omogućava brzu izradu modela, olakšava produkciju (treniranje i testiranje), te daje dobru platformu za dalje istraživanje [19,44].

4.1.6 Keras

Keras je sučelje za programiranje aplikacija koje nam nam omogućava korištenje algoritama dubokog učenja [19, 51]. Napisan je u Pythonu, a možemo ga definirati kao infrastrukturni sloj iznad TensorFlow paketa.

4.1.7 NumPy

NumPy je jedan od važnijih paketa za programski jezik Python [43]. Puno se koristi za znanstveno računanje. Napisan je u C-u. To je ”open-source” projekt koji daje efikasnu implementaciju n-dimenzionalnih listi (engl. NumPy vectorization). Također, daje matematičke alate poput generiranja nasumičnih brojeva, linearnu algebru, Fourierove transformacije, itd.

4.1.8 Scikit-learn

Scikit-learn je programska bibliteka za strojno učenje [28]. Daje nam jednostavne i efikasne alate za veliki broj algoritama strojnog učenja čije se implementacije lako koriste. Možemo koristiti programe za klasifikaciju, regresiju, grupiranje, reduciranje dimenzionalnisti, selekciju i procjenu kvalitete modela te izdvajanje i normalizaciju podataka.

4.1.9 Matplotlib

Matplotlib je programski paket za stvaranje statičkih, animiranih i interaktivnih grafova i dijagrama u Pythonu [52].

4.1.10 Plotly

Plotly je grafička biblioteka za programski jezik Python [53]. Koristi se za crtanje dvodimenzionalnih i trodimenzionalnih grafova koji mogu biti interaktivni i prikazivati se na webu. Možemo npr. izraditi osnovne i statističke grafove, mape, slike koje sadrže konture, ili histograme.

4.2 Baze podataka

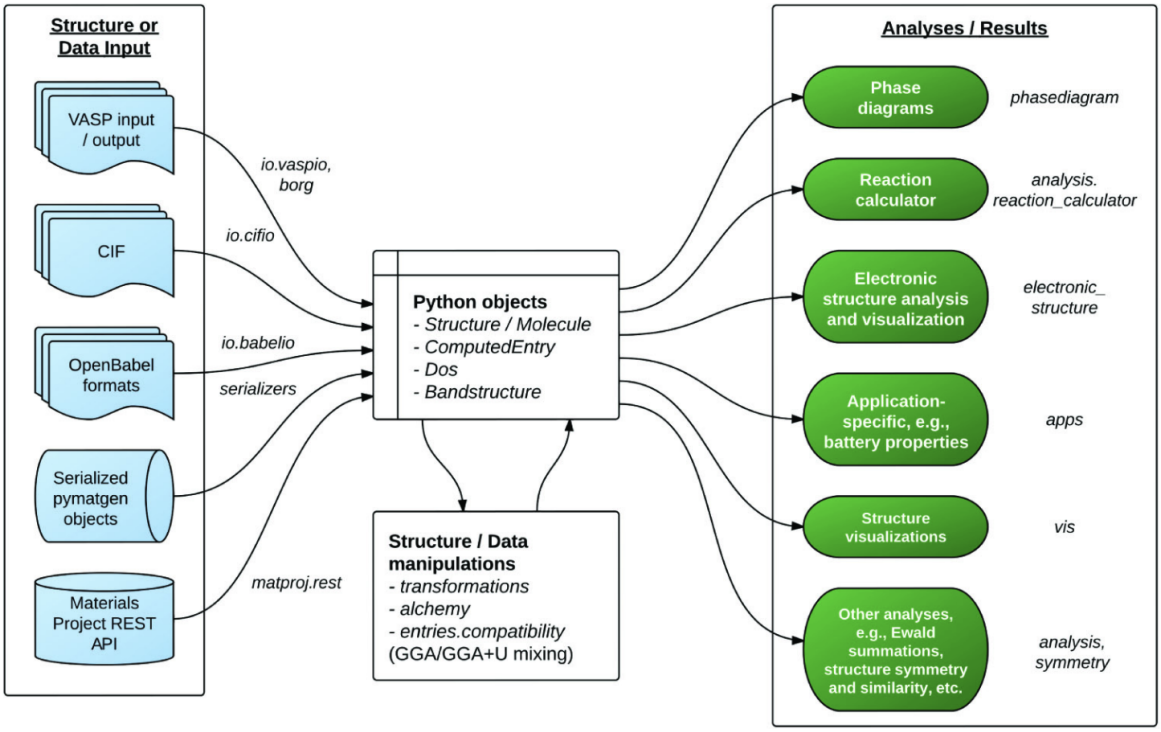
Baza podataka je organizirana zbirka strukturiranih podataka koji se elektronički pohranjuju u računalni sustav [54]. Danas se intenzivno radi na bazama koje sadrže informacije o osobinama materijala. U ovom radu koristimo baze Pymatgen [4, 5] i Mendeleev [6].

4.2.1 Pymatgen

Pymatgen (Python Materials Genomics) je programski paket otvorenog koda (engl. open source) za analizu materijala [4, 5]. Organiziran je na način da koristi fleksibilno dizajnirane klase koje predstavljaju objekte *Element, Molecule* i *Structure*. Pruža podršku za razne ulazno/izlazne formate, dobre alate za analizu, elektronsku strukturu materijala, fazne dijagrame, kemijske reakcije, itd... Pymatgen je dobro dokumentiran projekt koji sve više koriste znanstvenici.

Slika 4.1 predstavlja pregled tipičnog tijeka rada za korištenje Pymatgen baze. Korisnik nakon instanciranja baze koristi njezine funkcionalnisti da bi pretvorio podatke (strukture ili izračune) u Python objekte koji se koriste za daljnu analizu.

Pymatgen je strukturiran preko objektno orijentirane paradigme. Sastoji se od malog broja temeljnih paketa. Jedan od onih kojih smo koristili je *pymatgen.core.periodic_table*. Dio programskog koda koji koristi Pymatgen je prikazan na Slici 4.2. Koristivši referentni modul, pozvali smo objekt *Element* u čiji konstruktor šaljemo *String* varijablu koja predstavlja element iz periodnog sustava. Objekt potom poziva bilo koju od dostupnih *getter* funkcija (*thermal_conductivity, boiling_point, critical_temperature, brinell_hardness,...*), čije se povratne vrijednosti spremaju u listu koja će biti servirana modelu.



Slika 4.1: Pregled mogućeg tijeka rada za Pymatgen [4]

```
#svrstavanje elemenata u liste po pripadnim kristalnim strukturama
fcc_elements = ["Ag", "Al", "Au", "Cu", "Ir", "Ni", "Pb", "Pd", "Pt",
    "Rh", "Th", "Yb"]
bcc_elements = ["Ba", "Ca", "Cr", "Cs", "Eu", "Fe", "Li", "Mn", "Mo",
    "Na", "Nb", "Rb", "Ta", "V", "W" ]
hcp_elements = ["Be", "Cd", "Co", "Dy", "Er", "Gd", "Hf", "Ho", "Lu",
    "Mg", "Re",
                "Ru", "Sc", "Tb", "Ti", "Tl", "Tm", "Y", "Zn", "Zr"]

#konkatenacija
elements = fcc_elements + bcc_elements + hcp_elements

#getter funkcije klase Element
querable_pymatgen = ["atomic_mass", "atomic_radius", "
    electrical_resistivity","molar_volume", "bulk_modulus", "
    youngs_modulus",
                     "average_ionic_radius", "density_of_solid", "
    coefficient_of_linear_thermal_expansion"]

for item in elements:
    element_values = []
    #vrsimo upit nad bazom pymatgen (alias pymat) i dohvacamo element
    iz liste elements
    element_object = pymat.Element(item)
    #dohvacenom elementu pridodajemo sva svojstva koja smo definirali
    u querable_pymatgen
    for i in querable_pymatgen:
        element_values.append(getattr(element_object,i))

    #Kao rezultat dobivamo dvostruku listu koja ce nam posluziti kao "
    ulaz" za model
    all_values.append(element_values)
```

Slika 4.2: Programski kod koji koristi Pymatgen

4.2.2 Mendeleev

Mendeleev je programski paket koji nam daje sučelje pomoću kojeg u programima možemo dobiti informacije o raznim svojstvima kemijskih elemenata u periodnom sustavu, njihovih iona i izotopa [6]. Također nam daje jednostavan način vizualizacije koji nam omogućuje stvaranje prilagođenih periodičnih tablica koje prikazuju različita svojstva. Paket je izgrađen preko objektno orijentirane paradigme, te instanciranjem temeljnih klasa pozivamo razne *getter* funkcije koje nam daju informacije o

različitim svojstima za zadani element.

4.3 Temperature taljenja metala: linearna regresija

U ovom radu linearnu regresiju iz scikit-learn paketa koristimo za ispitivanje mogućih ovisnosti između temperature taljenja i drugih stvojstava za skup metala. Svojstva koja ispitujemo su: Youngov modul elastičnosti, konstanta rešetke i koeficijent linearne termičke ekspanzije. Račun se izvršava u Jupyterovoj bilježnici koja je instalirana na NanoHubu [48].

U skupu za treniranje su metali: srebro, aluminij, zlato, iridij, nikal, olovo, paladij, rodij, torij, barij, krom, željezo, litij, molibden, natrij, niobij, volfram, berilij, kalcij, kadmij, kobalt, erbij, hafnij, holmij, lutecij, renij, rutenij, terbij, talij, tulij, itrij, cink, mangan, skandij, disprozij, kositar, platina, cerij, vanadij, cirkonij, bakar, europij i magnezij. U skupu za testiranje su metali: tantal, titanij, gadolinij, samarij, iterbij i bizmut. Za izračun regresijske linije korištena je linearna regresija najmanjih kvadrata. To je vrsta regresijske metode koja procjenjuje parametre u regresijskom modelu minimizirajući zbroj kvadratnih ostataka. Pri tome se minimizira zbroj kvadrata razlika između promatranih vrijednosti i odgovarajućih naučenih vrijednosti. Dio programskg koda za regresiju je prikazan na Slici 4.3.

```
def regression(x_train, x_test, y_train, y_test):

    # Definiramo model i treniramo ga
    model = linear_model.LinearRegression()
    model.fit(x_train, y_train)

    # Spajamo podatke za treniranje i testiranje
    full_x = np.concatenate((x_train, x_test), axis=0)
    full_y = np.concatenate((y_train, y_test), axis=0)

    # Koristimo model za predviđanje
    predictions = model.predict(full_x)

    # Ispis linearnog modela i informacija o greškama
    print("Linear Equation: %.4e X + (%.4e)"%(model.coef_, model.intercept_))
    print("Mean squared error: %.4e" % (mean_squared_error(full_y, predictions)))
    print('Variance score: %.4f' % r2_score(full_y, predictions))

    return predictions
```

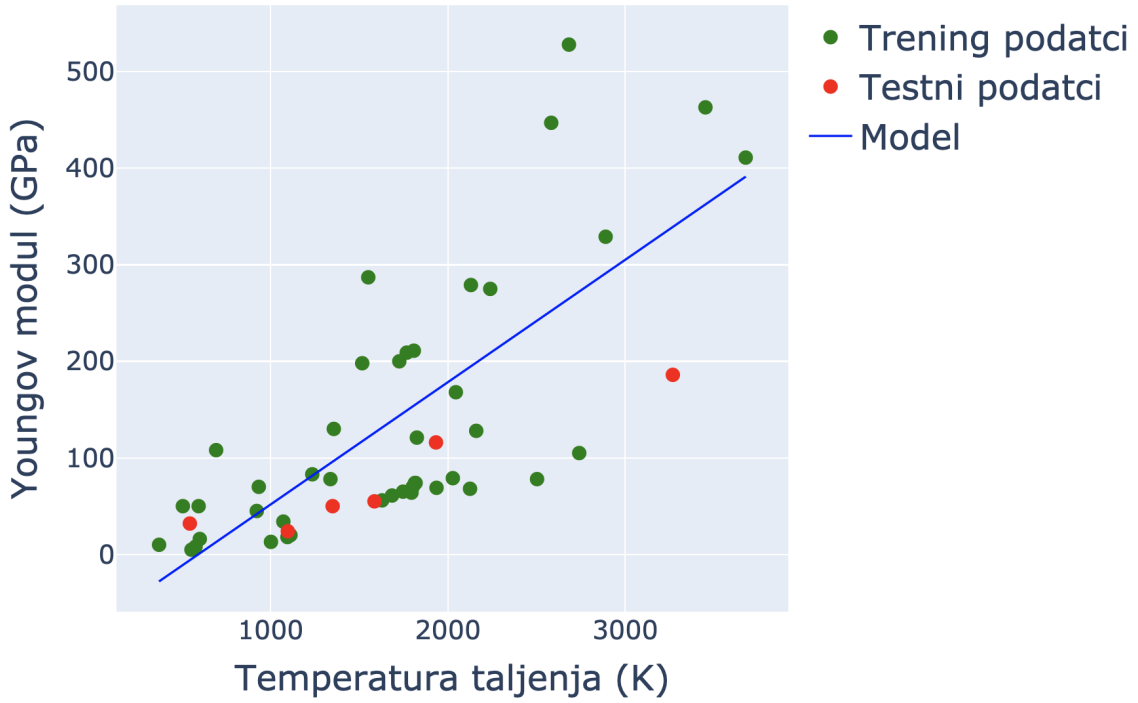
Slika 4.3: Implementacija regresijske funkcije u programskom jeziku Python

Ovisnost temperature taljenja o Youngovom modulu

Youngov modul elastičnosti definiran je omjerom naprezanja i deformacije, kao što je prikazano u jednadžbi (4.1) [55].

$$E = \frac{\sigma}{\epsilon} \tag{4.1}$$

Pri tome je E Youngov modul izražen u paskalima, σ jednosmjerno naprezanje, a ϵ bezdimenzionalna veličina koja opisuje promjenu duljine podijeljena s izvornom duljinom. Youngov modul se koristi za predviđanje kompresije ili istezanja linearnih predmeta poput žica i štapova sve dok je naprezanje manje od granice elastičnosti.



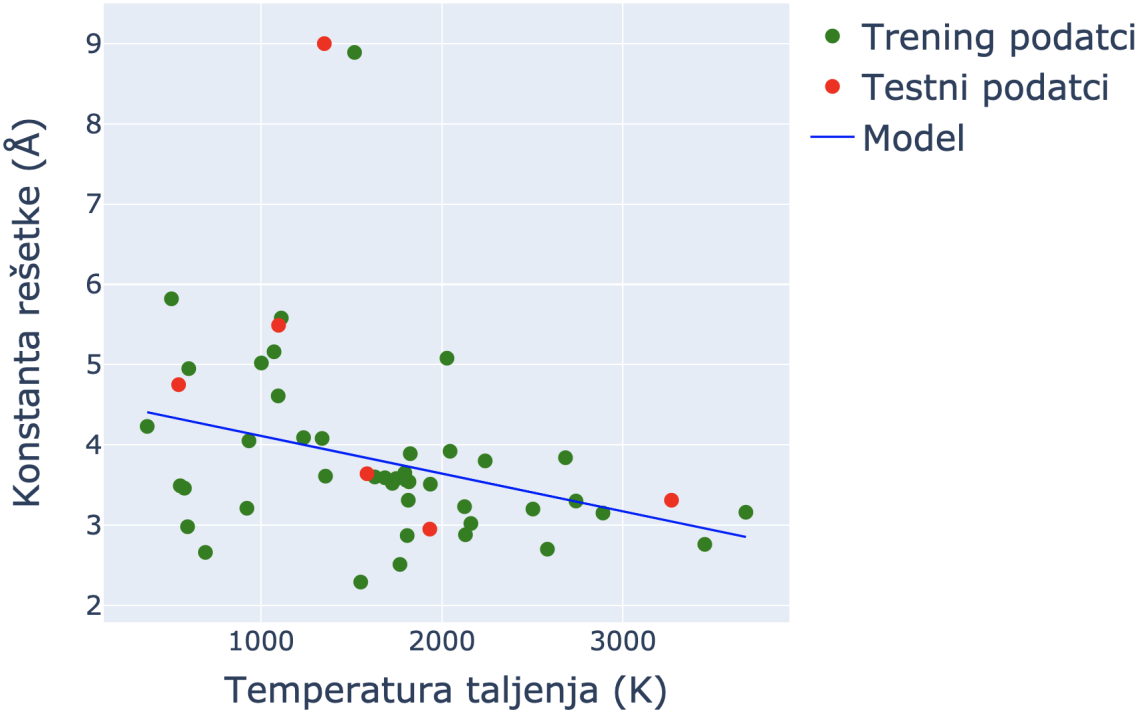
Slika 4.4: Ovisnost temeperature taljenja o Youngovom modulu elastičnosti

Slika 4.4 prikazuje rezultate linearnog regresijskog modela za ovisnost temperature taljenja o Youngovom modulu za metale iz skupova za treniranje i testiranje. Kod skupa za testiranje manja su odstupanja za titanij čija je temperatura taljenja $T_m = 1933$ K, gadolinij ($T_m = 1586$ K), samarij ($T_m = 1350$ K), iterbij ($T_m = 1097$ K) i bizmut ($T_m = 544.5K$). Veće odstupanje pokazuje tantal s temperaturom taljenja $T_m = 3269$ K. U skupu za treniranje od regresijske linije najviše odstupa iridij ($T_m = 2683$ K). Linearna regresija je opisana jednadžbom (4.2). Srednja kvadratna pogreška MSE (engl. mean squared error) je prikazana u (4.3), a varijanca σ^2 u (4.4).

$$Y = 1.2666 \times 10^{-1} X - 74.861 \tag{4.2}$$

$$MSE = 7.7780 \times 10^{3} \tag{4.3}$$

$$\sigma^2 = 0.5193 \tag{4.4}$$



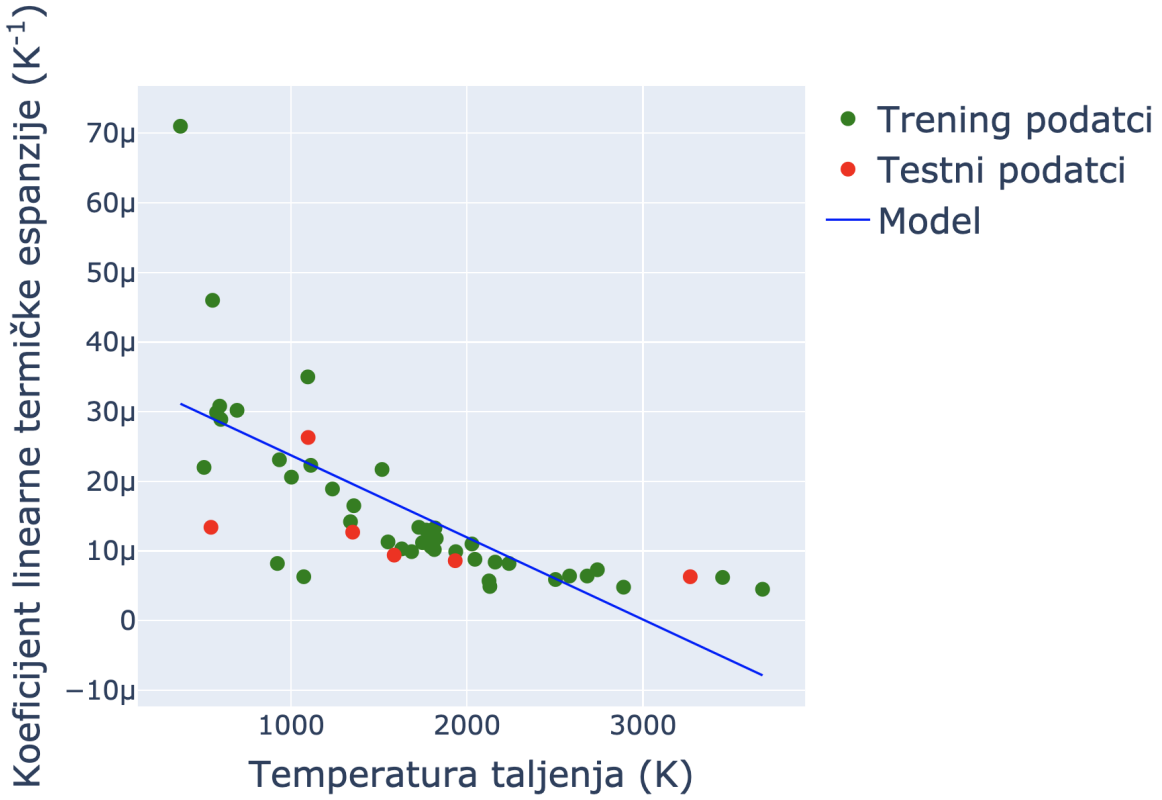
Slika 4.5: Ovisnost temeperature taljenja o konstanti rešetke

Ovisnost temeperature taljenja o konstanti rešetke Konstanta rešetke je parametar koji opisuje dimenzije jediničnih ćelija kristalne rešetke [56]. Slika 4.5 prikazuje regresijsku liniju za ovisnost temperature taljenja o konstanti rešetke. Iz skupa za testiranje tantal ($T_m = 3290$ K), titanij ($T_m = 1941$ K), gadolinij ($T_m = 1586$ K) i bizmut ($T_m = 544.6$ K) imaju najmanje odstupanje od regresijske linije, dok iterbij ($T_m = 1097$ K) pokazuje nešto veće odstupanje. Iz rezultata je lako prepoznati pozitivnu korelaciju temperature taljenja i konstante rešetke za sve metale iz skupova za treniranje i testiranje izuzev mangana ($T_m = 1519$ K) i samarija ($T_m = 1345$ K) koji pokazuju vrlo veliko odstupanje od regresijske linije. Regresijska jednadžba, srednja kvadratna pogreška i varijanca su prikazane u jednadžbama 4.5, 4.6 i 4.7.

$$Y = -4.6946 \times 10^{-4} X - 4.5810 \tag{4.5}$$

$$MSE = 1.5683 \tag{4.6}$$

$$\sigma^2 = 0.0969 \tag{4.7}$$



Slika 4.6: Ovisnost temeperature taljenja o koeficjentu linearne termičke ekspanzije pri čemu je $\mu = 10^{-6}$

Ovisnost temperature taljenja o koeficijentu linearne termičke ekspanzije Koeficijent linearne termičke ekspanzije je omjer relativnog širenja i promjene temperature kao što je opisano jednadžbom (4.8) [57].

$$\alpha_L = \frac{1}{L}\frac{dL}{dT} \tag{4.8}$$

Pri tome je α_L koeficijent linearne termičke ekspanzije izražen u $[K^{-1}]$, dL je infinitezimalna promjena duljine uzorka zbog zagrijavanja ili hlađenja, L je izvorna duljina uzorka na sobnoj temperaturi, a dT je infinitezimalna temperaturna promjena.

Slika 4.6 prikazuje ovisnost temperature taljenja o koeficijentu linearne termičke ekspanzije. Većina metala iz skupa za treniranje prati regresijsku liniju, ali su odstu-

panja velika za elemente koji imaju najveće i najmanje temperature taljenja. Specijalno, natrij ($T_m = 370.9$ K) koji se nalazi u skupu za treniranje jako odstupa od regresijske linije. Skoro svi metali iz skupa za testiranje uredno prate regresijsku liniju osim bizmuta ($T_m = 544.6$ K) koji pokazuje odstupanje. Regresijska linija je zadana jednadžbom 4.9. Srednja kvadratna pogreška i varijanca su prikazane u jednadžbama 4.10 i 4.11.

$$Y = -1.1793 \times 10^{-8} X - 3.5531 \times 10^{-5} \tag{4.9}$$

$$MSE = 7.6499 \times 10^{-11} \tag{4.10}$$

$$\sigma^2 = 0.4754 \tag{4.11}$$

4.4 Predviđanje strukture kristala metala: neuronske mreže

Analizirana je mogućnost predviđanja strukture kristalne rešetke metala uporabom neuronskih mreža. Korištena je Jupyterova bilježnica koja je instalirana na NanoHubu [48]. U skupu za treniranje su metali: srebro, aluminij, nikal, paladij, rodij, barij, krom, željezo, litij, molibden, natrij, niobij, volfram, berilij, kalcij, kadmij, kobalt, erbij, hafnij, holmij, lutecij, rutenij, terbij, tulij, itrij, cink, mangan, skandij, disprozij, rubidij, cezij, vanadij, cirkonij, bakar, europij, magnezij, tantal, titanij, gadolinij i iterbij. U skupu za testiranje su metali: iridij, platina, zlato, olovo, torij, renij i talij.

Na samom početku rješavanja problema klasifikacije, potrebno je prevesti tekstualnu reprezentaciju kristalnih struktura u numeričke vrijednosti. Jedna od mogućnosti takvog prevođenja je one hot kodiranje (engl. one hot encoding) [58]. U takvom kodiranju su kristalne strukture predstavljene vektorima s tri komponente, pri čemu je jedna od njih jedinica, a ostale su nula. Prevođenje za nekoliko metala je prikazano u Tablici 4.1.

Element	Kristalna struktura	One hot kodiranje
Ag	FCC	[1,0,0]
Li	BCC	[0,1,0]
Zn	HCP	[0,0,1]
Ni	FCC	[1,0,0]

Tablica 4.1: Primjer kodiranja za kristalne strukture

Sljedeći korak je normalizacija podataka. Normalizacija je tehnika koja se često primjenjuje kao dio pripreme podataka za strojno učenje. Cilj normalizacije je promi-

jeniti vrijednosti numeričkih stupaca u skupu podataka kako bi se koristila zajednička ljestvica, bez narušavanja razlika u rasponima vrijednosti ili gubitka podataka. Koristimo standardnu normalizaciju gdje se oduzima srednja vrijednost i dijeli sa standardnom devijacijom, kao sto je prikazano jednadžbom 4.12 [59].

$$Z = \frac{X - \mu}{\sigma} \tag{4.12}$$

U jednadžbi 4.12 podaci iz skupa za treniranje ili skupa za testiranje označeni su kao X. Aritmetička sredina tih podataka je označena kao μ, a standardna devijacija je σ.

Koristimo neuronsku mrežu u kojoj se prvi sloj sastoji od šesnaest čvorova, drugi sloj od dvanaest te treći od tri čvora. Mreža je prikazana na Slici 4.8. Za prva dva sloja koristimo *ReLU* (3.12), dok za izlazni sloj koristimo softmax (3.14) aktivacijsku funkciju. Ulazni sloj od šestnaest neurona je reprezentacija osamnaest fizikalnih svojstava koja su karakteristična za promatrane metale. Broj ulaznih varijabli ne mora nužno biti jednak broju neurona ulaznog sloja iz razloga što parametar aktivacijske funkcije može biti linearna funkcija više ulaznih varijabli. Te ulazne varijable su: atomski broj, atomski volumen, temperatura vrelišta, Ghosh-ova skala elektronegativnosti, temperatura taljenja, entalpija, konstanta rešetke, temperatura taljenja, specifična toplina, atomska masa, atomski radijus, električna otpornost, molarni volumen, volumni modul elastičnosti, Youngov modul, prosječni ionski radijus, gustoća krutine i koeficijent linearne termičke ekspanzije.

Nakon što je definirana struktura neuronske mreže, odgovarajući optimizator mora biti izabran kako bi model dao najbolje moguće rezultate. Optimizatori definiraju kako neuronske mreže uče, odnosno traže vrijednosti parametara za koje funkcija troška poprima najmanje vrijednosti. Osnovni primjeri optimizatora su *batch* (3.7) i *stohastički* gradijentni spust (3.8). Gradijenti vrlo složenih funkcija poput neuronskih mreža imaju tendenciju ili nestati, ili eksplodirati dok se podaci šire kroz funkciju. *RMSProp* je optimizator koji uravnotežuje veličinu koraka (zamah), smanjujući korak za velike gradijente kako bi se izbjeglo eksplodiranje, a povećavajući korak za male gradijente kako bi se izbjeglo nestajanje [60]. U suštini, RMSProp koristi prilagodljivu brzinu učenja umjesto da brzinu učenja tretira kao statički parametar. Iz tog razloga RMSProp optimizator je odabran u definiciji modela. Dio programskog koda koji definira strukturu neuronske mreže, koristi RMSProp optimizator i računa točnost je prikazan na Slici 4.7.

Početak klasifikacije je postupak prolaza unaprijed gdje se računa predviđanje rezultata i uspoređuje ga se sa stvarnom vrijednošću kako bi se dobila pogreška. Kako bi pogreška bila svedena na minimum, prolazi se mrežom unatrag i metodom algoritama gradijetnog spusta se određuje koliko svaki neuron doprinosi ukupnoj

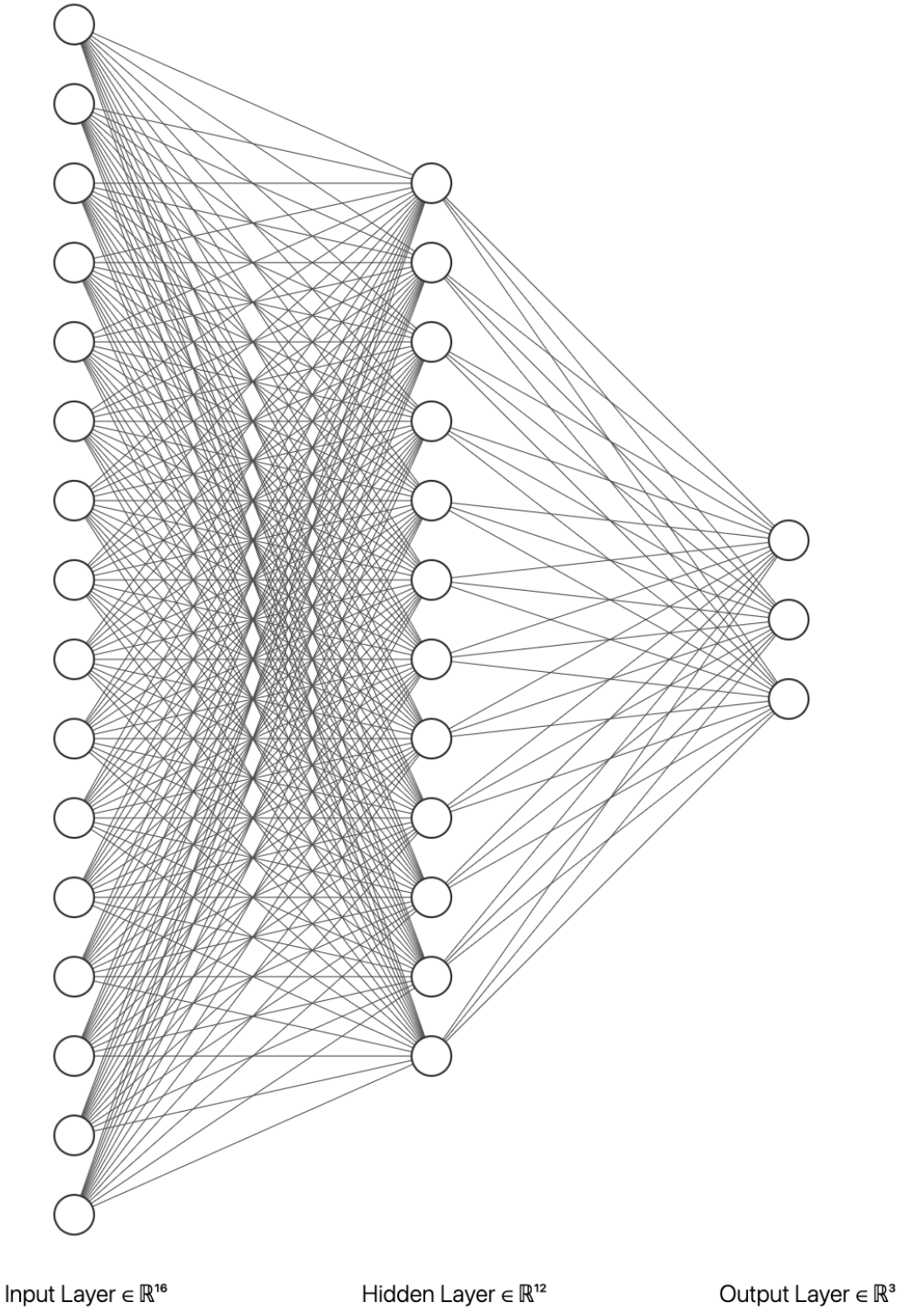
```
# DEFINICIJA MODELA

# Tezine ce biti nasumicno inicijalizirane, cilj je definirati model koji osigurava reproducibilnost
kernel_init = initializers.RandomNormal(seed=14)

model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(train_values.shape[1],), kernel_initializer=kernel_init))
model.add(Dense(12, activation='relu', kernel_initializer=kernel_init))
model.add(Dense(3, activation=tf.nn.softmax))  # Izlazni sloj

# DEFINICIJA OPTIMIZATORA

optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.002)

# Povezujemo model i optimizator, te definiramo metriku koja ce evaluirati tocnost modela
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
model.summary()
```

Slika 4.7: Programski kod koji određuje strukturu neuronske mreže

grešci. Izlazni sloj koristi softmax (3.14) aktivacijsku funkciju koja vraća pozitivne decimalne vrijednosti za svaki izlazni neuron. Suma svih vrijednosti je jednaka jedan, stoga se rezultati softmaxa interpretiraju kao distribucija vjerojatnosti. Za izlazni sloj kojeg čine tri neurona, aktivacijska funkcija pridružuje rezultantnu vrijednost svakom neuronu. Promatranjem vrijednosti svih neurona uzima se ona najveća, te se deklarira kao rezultat predviđanja kristalne strukture. Korištenje softmax aktivacijske funkcije daje mogućnost definiranja funkcije gubitka poznate kao ”gubitak unakrsne entropije” (engl. cross-entropy loss) koja mjeri razliku izmedju točnih predefiniranih vrijednosti i distribucije vjerojatnosti koja je predviđena od strane neuronske mreže [61].

Tablica 4.2 prikazuje klasifikaciju kristalnih struktura kao rezultat primjene neuronske mreže. Predviđene kristalne strukture dobivene su uzimanjem maksimalne vrijednosti svih izlaznih neurona za dani metal. Neuronska mreža je točno predvidjela gotovo sve strukture osim europija, koji ima BCC strukturu, no predviđena mu je HCP struktura. Slika 4.9 pokazuje distribuciju vjerojatnosti pojedinih metala za zadane kristalne strukture. Stupci označeni sa ”*” predstavljaju metale iz skupa za testiranje. Za FCC metale vidimo da je pridružena vjerojatnost za gotovo sve elemente 100%, osim za iridij i olovo čije pridružene vjerojatnosi iznose 99%. Za BCC metale su dobiveni slični rezultati. Blago odstupanje pokazuju tantal i željezo. Najveću pridruženu vjerojatnost za tantal ima BCC kategorija sa pripadnim rezultatom od 85.46%, dok FCC i BCC kategorije imaju slijedom pridružene vjerojatnosti od 12.14% i 2.40%. Kod željeza BCC kategorija ima pridruženu vjerojatnost od 98.4%, a vjerojatnost od 1.6% pridjeljena je HCP kategoriji. Europij ima pridruženu pogrešku od 100%, koji je ujedno i jedini pogrešno klasificiran metal. Rezultati za HCP metale pokazuju da blago odstupanje postoji samo kod magnezija čija HCP kategorija ima pridruženu vjerojatnost od 95.68%, dok su za BCC i HCP kategorije

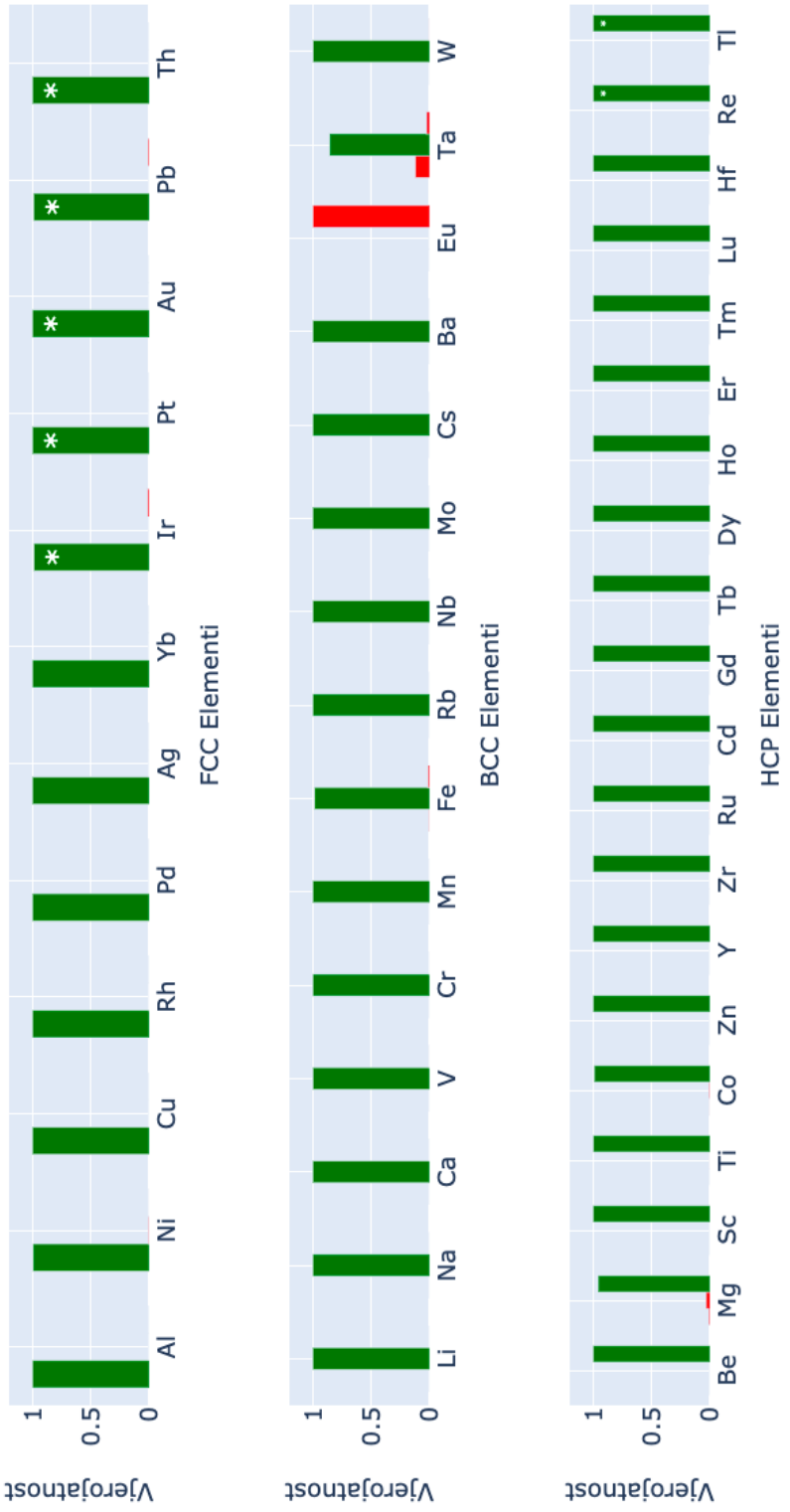
rezultati 3.27% i 1.05%. Postignuta točnost na skupu metala za testiranje je 85.71%.



Slika 4.8: Prikaz neuronske mreže korištene za klasifikaciju kristalnih struktura

Kemijski simbol	Atomski broj	Poznata kristalna struktura	Predviđena kristalna struktura
Co	27	HCP	HCP
Tm	69	HCP	HCP
Y	39	HCP	HCP
Re	75	HCP	HCP
Ni	28	FCC	FCC
Ho	67	HCP	HCP
Au	79	FCC	FCC
Sc	21	HCP	HCP
Rh	45	FCC	FCC
W	74	BCC	BCC
Gd	64	HCP	HCP
Tb	65	HCP	HCP
Hf	72	HCP	HCP
Yb	70	FCC	FCC
Cs	55	BCC	BCC
Zn	30	HCP	HCP
Ba	56	BCC	BCC
Mn	25	BCC	BCC
Fe	26	BCC	BCC
Mo	42	BCC	BCC
Na	11	BCC	BCC
Lu	71	HCP	HCP
Th	90	FCC	FCC
Cu	29	FCC	FCC
Li	3	BCC	BCC
Tl	81	HCP	HCP
V	23	BCC	BCC
Rb	37	BCC	BCC
Zr	40	HCP	HCP
Cr	24	BCC	BCC
Nb	41	BCC	BCC
Ag	47	FCC	FCC
Be	4	HCP	HCP
Ru	44	HCP	HCP
Al	13	FCC	FCC
Ti	22	HCP	HCP
Pb	82	FCC	FCC
Ca	20	BCC	BCC
Ta	73	BCC	BCC
Dy	66	HCP	HCP
Cd	48	HCP	HCP
Er	68	HCP	HCP
Pd	46	FCC	FCC
Eu	63	BCC	HCP
Ir	77	FCC	FCC
Mg	12	HCP	HCP
Pt	78	FCC	FCC

Tablica 4.2: Klasifikacija kristalnih struktura kao rezultat primjene neuronske mreže



Slika 4.9: Rezultati klasifikacijskog modela

5 Zaključak

Primjenom linearne regresije, metode nadziranog strojnog učenja, ispitana je ovisnost temperature taljenja o Youngovom modulu elastičnosti, konstanti rešetke i koeficijentu linearne termičke ekspanzije. Kod ovisnosti temperature taljenja o Youngovom modulu podaci iz skupova za treniranje i testiranje prate izračunatu liniju, ali postoje i velika odstupanja. Najviše odstupaju iridj iz skupa za treniranje i tantal iz skupa za testiranje. Kod ovisnosti temperature taljenja o konstanti rešetke najveća odstupanja od izračunatog linearnog modela pokazuju mangan iz skupa za treniranje i samarij iz skupa za testiranje. Kod ovisnosti temperature taljenja o koeficijentu linearne termičke ekspanzije može se zaključiti da i pored toga što većina podataka prati izračunati linearni model, velika odstupanja za najmanje i najveće temperature pokazuju da bi nelinearni model bolje opisivao podatke.

Istražena je mogućnost predviđanja strukture osnovnog stanja kristala metala primjenom nadziranog strojnog učenja. Korištena je metoda neuronskih mreža. Dobivena je točnost od 85.71%, što je zadovoljavajući rezultat za primjene strojnog učenja. Rezultati na odabranim skupovima za treniranje i testiranje su pokazali da jedino europij nema dobro predviđenu strukturu.

6 Metodički dio

Strojno učenje je danas veoma popularan termin u znanosti, ali sve više i u svakodnevnom životu. Za očekivati je da će to pobuditi interes učenika u srednjim školama. U ovom poglavlju izlažemo metodički pristup strojnom učenju za učenike srednjih škola. Strojno učenje je interdisciplinarno područje i zahtijeva određen nivo matematičke vještine, no za sredjoškolske potrebe možemo se fokusirati na konceptualnim i praktičnim primjenama.

Preporuka je da se ova tema obrađuje pred kraj srednjoškolskog obrazovanja nakon što učenici usvoje osnove programiranja u Pythonu te osnovni matematički aparat. U dva školska sata mogu se uvesti osnovni pojmovi. Primjer iz prakse se može predložiti za grupni rad.

6.1 Primjer sortiranja u spremnike za smeće

Započinjemo sat s predstavljanjem problema. Predlažemo učenicima da zamisle da su zaposleni u nekoj tvrtki. Zadano im je da stvore sistem koji će sortirati odbačene predmete u spremnike za smeće [62, 63]. Sistem mora identificirati plastiku, staklo i kartonske kutije, te ih automatski razvrstati u odgovarajuće spremnike. Ovim smo učenike potaknuli na razmišljanje i raspravu. Učenici zapisuju u bilježnicu svoja razmatranja, te potom profesor uvodi teorijsku podlogu za rješavanje problema, a to je grupiranje podataka algoritmom *k-sredina* (engl. k-means).

6.2 Algoritam k-sredina

Algoritam k-sredina je jednostavna i efikasna metoda nenadziranog strojnog učenja [18,19]. Algoritmi nenadziranog učenja uočavaju uzorke u skupu podataka bez pozivanja na poznate ishode. Modelu se ne govori što mora naučiti, već mu se dopušta da pronalazi uzorke i izvlači zaključke iz neobilježenih podataka. Zadaci učenja bez nadzora obično uključuju grupiranje sličnih primjera, smanjenje dimenzionalnosti i procjenu gustoće [19]. Algoritam k-sredina pretražuje neobilježeni višedimenzionalni skup podataka. Postoje dva koraka. U prvom koraku algoritam dodjeljuje sve točke podataka središtima najbližih klastera. U drugom koraku se postavljaju nova središta klastera kao srednja vrijednost svih točaka koje su dodijeljene središtima u prethodnom koraku. Algoritam je riješio zadatak kada se završi dodjela svih točaka i više nema promjena [18].

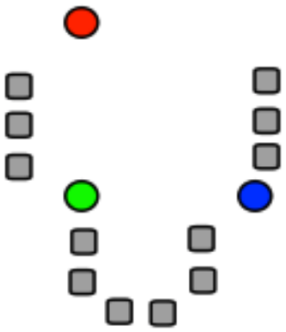
6.3 Rješavanje problema

Algoritmu moramo dati ulazne podatke. U općem slučaju algoritam k-sredina sam nalazi broj grupa k koje želimo formirati, no u ovom primjeru zadajemo taj broj. Imamo tri različita spremnika te je $k = 3$. Na ploču zapisujemo korake predstavljene na Slikama 6.1-6.5 [63]. Svaki kružić ispunjen bojom predstavlja odgovarajući spremnik za staklo, plastiku, ili kartonske kutije, dok kvadratići predstavljaju primjerke smeća. Prazni kružići predstavljaju srednju vrijednost svakog središta, koja se definira kao srednja vrijednost točaka iste boje. U prvom koraku nasumično biramo k središta, zatim računamo euklidsku udaljenost između svake točke i središta. Za učenike moramo uvesti pojam euklidske udaljenosti. To je rastojanje između dvije točke u euklidskom prostoru predstavljeno jednadžbom 6.1 [64].

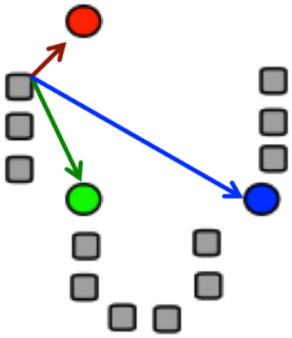
$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{6.1}$$

U jednažbi 6.1 x i y su dvije točke u euklidskom prostoru, x_i, y_i su euklidski vektori koji imaju zajedničko ishodište, a n predstavlja dimenziju prostora. Učenicima se mora naglasiti da se udaljenosti odnose na apstraktni prostor točaka koje predstavljaju spremnike i odgovarajuće otpatke. Nakon što smo završili s izračunom, dodjeljujemo svaku točku najbližem središtu. U sljedećem koraku, preračunavamo položaj svakog središta kao srednju vrijednost točaka istih boja. Središta se pomjeraju na novu lokaciju. Proces se ponavlja sve dok se središta ne prestanu pomjerati [63].

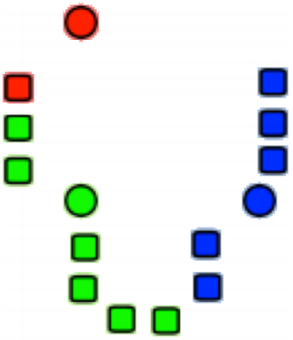
Nakon što smo postavili teorijsku podlogu, učenike razvrstavamo u grupe po četiri te predstavnik svake grupe na kraju sata predstavlja konceptualno rješenje. Jedno od rješenja može biti transportna traka preko koje se spremnici pomiču. Senzorima se mjeri njihova neprozirnost, a zatim se pomoću vage određuje njihova težina. Neprozirnost i težina nam predstavljaju ulazne varijable koje šaljemo algoritmu na obradu. Algoritam daje tri skupine koje odgovaraju za tri različite vrste spremnika. Kad određeni spremnik dođe na kraj pokretne trake, sustav automatski pokreće odgovarajuću kantu ispod spremnika na temelju grupiranja koje je odredio algoritam.



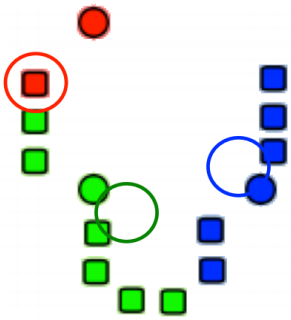
Slika 6.1: Nasumično se bira k središta



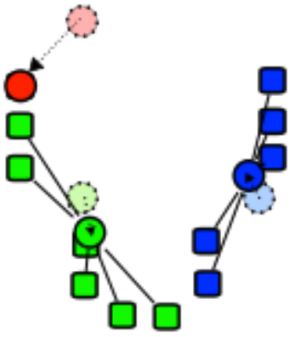
Slika 6.2: Izračunava se euklidska udaljenost između svakog kvadratića i svih središta



Slika 6.3: Svaki kvadratić se dodjeljuje najbližem središtu



Slika 6.4: Preračunava se položaj svakog središta kao srednja vrijednost za kvadratiće iste boje



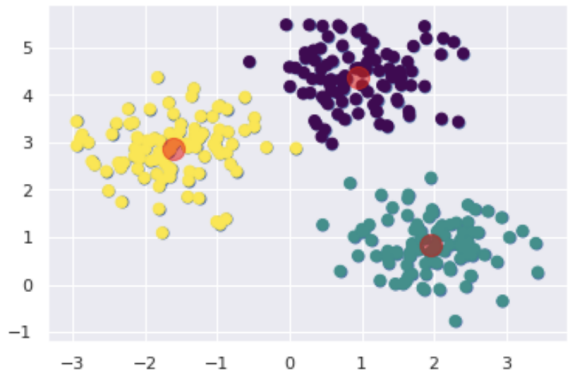
Slika 6.5: Pomjeraju se središta na novu lokaciju

Za potrebe rješavanja zadatka predstavljamo učenicima biblioteku scikit-learn i njenu implementaciju algoritma k-means [28, 65]. Učenicima objašnjavamo parametre potrebne za inicijalizaciju modela i načine generiranja nasumičnih podataka. Za domaću zadaću zadajemo programski kod s adekvatnim komentarima. Učenici moraju nadopuniti taj kod da bi dobili konačno rješenje. Dio programskog koda je

predstavljen na Slici 6.6. Grafičko rješenje zadatka sortiranja smeća u tri spremnika je prikazano na Slici 6.7.

```
#Stvaramo podatke koji nam predstavljaju predmete smeca
X, y_true = make_blobs(n_samples=300, centers=3,
                       cluster_std=0.60, random_state=0)
#ispis tih podataka
plt.scatter(X[:, 0], X[:, 1], s=50);

####### Zadaca za ucenike #########

#Inicijaliziramo KMeans algoritam kojo automatski postavlja tocke clustera
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

###################################


plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5);
```

Slika 6.6: Pozivanje algoritma k-sredina iz biblioteke *scikit-learn*



Slika 6.7: Prikaz rješenja zadatka

Literatura

[1] Mitchell, T. M. Machine Learning. New York: McGraw-Hill, 1997.

[2] Schmidt, J.; Marques, M. R. G.; Botti, S.; Marques, M. A. L. Recent advances and applications of machine learning in solid-state materials science//npj Computational Materials. Vol. 5, (2019), 83.

[3] Wei, J. et al. Machine learning in materials science // InfoMat. Vol. 1, (2019), str. 338-358.

[4] Pymatgen, `https://pymatgen.org/`, 20. 12. 2020.

[5] Ong, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. // Computational Materials Science. Vol. 68, (2013), str. 314-319.

[6] Mendeleev, `https://tinyurl.com/y5aztltk`, 4. 12. 2020.

[7] nanoHUB, `https://nanohub.org`, 23. 2. 2021.

[8] Kittel, C. Introduction to Solid State Physics, 8th ed. Hoboken, NJ: John Wiley & Sons, 2005.

[9] Mahan, G. D.; Douglas, R. W.; Zallen, R. Amorphous solid, Encyclopedia Britannica, `https://tinyurl.com/z9yls64`, 2. 12. 2020.

[10] Atkins, P.; de Paula, J. Elements of Physical Chemistry, 7th ed. Oxford, UK: Oxford University Press, 2017.

[11] Šips, V. Uvod u fiziku čvrstog stanja, 2.izdanje. Zagreb: Školska knjiga

[12] Batistić, I. Kristalna struktura (prezentacija predavanja), `https://tinyurl.com/y33fkab6`, 15. 12. 2020.

[13] Wigner-Seitzova ćelija, `https://tinyurl.com/y4229m7m`, 16. 12. 2020.

[14] Bravais lattice, Wikipedia, `https://tinyurl.com/nj824tk`, 16. 12. 2020.

[15] The Arrangement of Atoms in Crystalline Solids, `https://tinyurl.com/y6bcrw3l`, 17. 12. 2020.

[16] Hexagonal Close Packed Crystal Structure, `https://tinyurl.com/y545vrtv`, 17. 12. 2020.

[17] Bonaccorso, G. Mastering Machine Learning Algorithms. Birmingham: Packt Publishing, 2020.

[18] Müller, A. C; Guido, S. Introduction to Machine Learning with Python. 1st ed. Sebastopol: O'Reilly Media, 2017.

[19] Geron, A. Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow. 2nd ed. Sebastopol: O'Reilly Media, 2019.

[20] Osinski, B.; Budek, K. What is reinforcement learning? The complete guide. https://tinyurl.com/y28mpc8n, 20. 12. 2020.

[21] The Wikipedia Guide, Introduction to Machine Learning, https://tinyurl.com/yb453twf, 14. 12. 2020.

[22] Ng, A; Lecture Notes, http://cs229.stanford.edu/notes2020spring/cs229-notes1.pdf, 15. 12. 2020.

[23] Liu, C. T.; Militon, J.; McIntosh, A. Correlation and Regression with R, https://tinyurl.com/yylcqjyk, 15. 12. 2020.

[24] How Simple Linear Regression Works, https://tinyurl.com/y5vxd3fy, 16. 12. 2020.

[25] McDonald, C. Machine learning fundamentals (I): Cost functions and gradient descent, https://tinyurl.com/y5mgww7z, 17. 12. 2020.

[26] Stohastic Gradient Descent, https://tinyurl.com/1duigfuj, 17. 12. 2020.

[27] SciPy.org, https://www.scipy.org, 4. 2. 2021.

[28] Scikit-learn, https://scikit-learn.org/stable/, 4. 2. 2021.

[29] VanderPlas, J. Linear Regression. https://tinyurl.com/3zdfv6lk, 21. 2. 2021.

[30] Robinson, S. Linear Regression in Python with Scikit-Learn, https://tinyurl.com/2bpass8z, 21. 2. 2021.

[31] Stojiljković, M. Linear Regression in Python, https://tinyurl.com/19piov4b, 21. 2. 2021.

[32] Fridman, L. MIT Deep Learning Basics: Introduction and Overview with TensorFlow, https://tinyurl.com/yb4knhmx, 23. 12. 2020.

[33] Lefkowitz, M. Professor's perceptron paved the way for AI – 60 years too soon, https://tinyurl.com/y4qchvbv, 23. 12. 2020.

[34] Wikipedia, Punched card, https://tinyurl.com/letncxn, 23. 12. 2020.

[35] DeepAI, Weight (Artificial Neural Network), https://tinyurl.com/y56k2ez3, 3. 1. 2021.

[36] Parmezan, A. R. S.; Souza, V. M. A; Batista, G. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model.// Information Sciences. Vol. 484 (2019), str. 302-337.

[37] Patel, A. Tools to Design or Visualize Architecture of Neural Network, https://tinyurl.com/yyqmofok, 8. 1. 2021.

[38] Nwankpa, C. E.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning, https://tinyurl.com/y34jlsdp, 9. 1. 2021.

[39] Sigmoid function, https://tinyurl.com/1gwpsjto, 20. 12. 2020.

[40] Wood, T. What is the Softmax Function?, https://tinyurl.com/ks6tre4q, 10. 1. 2021.

[41] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Leaning Internal Representations by Error Propagation, ICS Report 8506, Institute for Cognitive Science, University of California, San Diego (1985), 8. 1. 2021.

[42] Sporici, D. C# Backpropagation Tutorial (XOR), https://tinyurl.com/3zxdosvy, 10. 2. 2021.

[43] NumPy, https://numpy.org/, 8. 1. 2021.

[44] TensorFlow, https://www.tensorflow.org/, 8. 1. 2021.

[45] PyTorch, https://pytorch.org, 8.1. 2021.

[46] Sarkar, T. Why you should forget 'for-loop' for data science code and embrace vectorization, https://tinyurl.com/ycasqvdf, 8. 1. 2021.

[47] Python, http://www.python.org, 9. 1. 2021.

[48] Gastelum, J. C. V.; Strachan, A.; Desai, S. Machine Learning for Materials Science, https://nanohub.org/resources/mseml, 22. 2. 2021.

[49] Project Jupyter, https://jupyter.org/, 9. 1. 2021.

[50] pandas, https://pandas.pydata.org/, 9. 1. 2021.

[51] Keras, https://keras.io/, 9. 1. 2021.

[52] Matplotlib, https://matplotlib.org/, 9. 1. 2021.

[53] Plotly Python Open Source Graphing Library, https://plotly.com/python/, 9. 1. 2021.

[54] Oracle, What is a Database, https://tinyurl.com/y3cplopm, 10. 1. 2021.

[55] HyperPhysics, Young's Modulus, http://hyperphysics.phy-astr.gsu.edu/hbase/permot3.html, 24. 2. 2021.

[56] Wikipedia, Lattice constant, https://en.wikipedia.org/wiki/Lattice_constant, 24. 2. 2021.

[57] NETZSCH Group, Coefficient of Linear Thermal Expansion (CLTE/CTE), https://tinyurl.com/23m84ue7, 24. 2. 2021.

[58] Brownlee, J. Why One-Hot Encode Data in Machine Learning?, https://tinyurl.com/6ru7tv9h, 28. 2. 2021.

[59] DeepAI, What is a Z-score, https://tinyurl.com/k249ut79, 26. 2. 2021.

[60] DeepAI, What is RMSPRop?, https://tinyurl.com/475ttrrm, 26. 2. 2021.

[61] Koech, K. E. Cross-Entropy Loss Function, https://tinyurl.com/7vy39ktb, 28. 2. 2021.

[62] Essinger, S.; Rosen, G. L. An introduction to machine learning for students in secondary education, https://tinyurl.com/y39n4k63, 13. 1. 2021.

[63] Gordon, J. B. Machine Learning Exercises for High School Students, https://tinyurl.com/ycvg73xo, 13. 1. 2021.

[64] Wikipedia, Euclidean distance, https://tinyurl.com/56kxexup, 26. 2. 2021.

[65] Scikit-learn: KMeans, https://tinyurl.com/6yzfu6tb, 28. 2. 2021.