Izrada modela za procjenu rizika od raka prostate

Perić, Mateo

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

MATEO PERIĆ

IZRADA MODELA ZA PROCJENU RIZIKA OD RAKA PROSTATE

ZAVRŠNI RAD

ČAKOVEC, 2023.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

MATEO PERIĆ

IZRADA MODELA ZA PROCJENU RIZIKA OD RAKA PROSTATE

PRODUCTION OF PROSTATE CANCER RISK ASSESSMENT MODEL

ZAVRŠNI RAD

Mentor:

mr. sc Željko Knok, v. pred.

ČAKOVEC, 2023.

ZAHVALA

Zahvaljujem svim profesorima i kolegama na uloženom trudu i vremenu tijekom studiranja na Međimurskom veleučilištu.

Posebna zahvala mojem mentoru mr. sc. Željku Knoku na strpljivosti i usmjeravanju u novi svijet umjetne inteligencije i strojnog učenja te na svim stručnim savjetima koji su mi na ovaj ili onaj način pomogli u izradi ovog rada.

Na kraju, posebne zahvale mojoj obitelji koja me neumorno vodila i bodrila kroz obrazovanje.

Mateo Perić

SAŽETAK

Tema ovog završnog rada je izrada modela za procjenu rizičnosti raka prostate na osnovu biopsije prostate. Rad je namijenjen promicanju umjetne inteligencije kako u medicinskoj sferi tako i u ostalim sferama života. Rad je zasnovan na prednostima i nedostatcima dijagnostike izvršene od strane umjetne inteligencije te ukazuje na odstupanja u rezultatima s obzirom od koje institucije slike dolaze tj. ovisno o kvaliteti slike koju koristimo.

Model je razvijen u sklopu programskog jezika Python. Za razvojno okruženje korištena je MiniConda. Programski kod napisan je u Jupyter Notebook. Implementacija neuronskih mreža i treniranje modela izvršeno je koristeći Tensorflow i Keras modula. U početku ovog rada je pojašnjeno na koji način se dolazi do biopsije prostate, kako se vrednuje agresivnost raka prostate te su pojašnjene ISUP skala te Gleason vrijednost po kojima se vrednuje agresivnost. Nakon toga su pojašnjene tehnologije koje su korištene za izradu i testiranje modela s velikim naglaskom na programski jezik Python te programske okvire bez kojih treniranje modela ne bi bilo moguće Tensorflow i Keras.

Pripremanje podataka i učenje modela obrađeno je u praktičnom dijelu rada kroz Jupyter Notebook kao razvojna okolina u jeziku Python uz upotrebu okvira Tensorflow-a i Keras-a.

Korisničko sučelje je implementirao u vidu macOS aplikacije, koristeći programski jezik Swift te kroz korištenje biblioteka za pokretanje programskog koda Python u Swiftu. Vizualni prikaz implementiran je koristeći SwiftUI.

Ključne riječi: Python, Keras, Tensorflow, MiniConda, Jupyter Notebook, ISUP Grade, Gleason Score, Swift, SwiftUI, Python

SADRŽAJ

1. UVOD

Umjetna inteligencija već dugi niz godina pokušava prodrijeti u sve dijelove života. Personalizirane reklame, personaliziran sadržaj na društvenim mrežama samo su neke od posljedica korištenja umjetne inteligencije. Uz dovoljnu količinu podataka i dovoljno veliku računalnu moć, umjetna inteligencija može postići rezultate koje su nedavno bili izvan dohvata ljudskog razuma. Od predviđanja ponašanja pojedinca do prognoza raznoraznih bolesti, može se reći da je ovo tek početak izuzetno nagle i čovjeku neshvatljive promjene koja uz veliku mogućnost donošenja pozitivnih promjena sa sobom donosi i neke negativne promjene. Umjetna inteligencija je već dugi niz godina u medicinskoj sferi s namjerom olakšavanja dijagnoza i prevencije razno raznih bolesti. Kod donošenja dijagnoza vezanih uz slike tkiva doktori koriste prvobitno vid za pronalazak nepravilnosti kod tkiva te na osnovu tih nepravilnosti donose dijagnoze. Računalo ima mogućnost visoke apstrakcije slikovnih podataka koje ljudsko oko ne može uočiti ni pojmiti te se zbog toga sve više pokušava koristiti umjetna inteligencija kod klasificiranja slika koje imaju mogućnost visokog nivoa apstrakcije.

2. CILJ I DOPRINOS RADA

Cilj ovog rada je potaknuti upotrebu umjetne inteligencije u medicini odnosno u granama dijagnosticiranja bolesti. Omogućiti bržu i točniju dijagnozu brzo napredujućih bolesti kao što su rak prostate, pluća i dojke te prikazati nedostatke kod korištenja umjetne inteligencije kod dijagnoza. Model bi služio kao dodatna provjera kod dijagnosticiranja raka od strane medicinskog osoblja. Model je implementiran pomoću konvolucijskih neuronskih mreža kroz programske okvire kao što stu Tensorflow i Keras. Vizualizacija podataka izvršena je uz Matplotlib biblioteke s glavnim ciljem da se prikaže jednostavnost stvaranja modela za učenje te uputi na moguće greške tokom izvedbe istog. Model je implementiran kroz macOS aplikaciju koja je dostupna svima za izvršavanje predikcija.

3. KORIŠTENE TEHNOLOGIJE

U ovom poglavlju opisane su tehnologije pomoću kojih je implementiran praktični dio završnog rada, s naglaskom na programski jezik Python. Detaljno su opisani Tensorflow i Keras zbog kojih je treniranje modela dostupno skoro svima, te ostali okviri uz koje je proces učenja i vizualizacije podataka olakšan.

3. 1. Python

Python je objektno orijentirani programski jezik visokog interpretacijskog nivoa. Uz to jedan je od najpopularnijih programskih jezika zbog jednostavnosti pisanja koda. Uz to što je objektno orijentirani jezik, Python podržava i pisanje funkcijskog koda. Uspješno povezuje razdaljinu između poslovnih programskih rješenja i programerskih mogućnosti. Svoju popularnost duguje, u jednu ruku, umjetnoj inteligenciji i neuronskim mrežama koje su trenutno u „procvatu“. Smatra se da je jedan od najboljih i najjednostavnijih programskih jezika za početnike koji žele zakoračiti u svijet programiranja i umjetne inteligencije.

Python je svoju primjenu pronašao u skoro svim oblastima računarstva od sistemskog programiranja do weba. Njegovi počeci sežu u kasne 80-te, a još se i danas razvija i unaprjeđuje.

3. 1. 1 Biblioteke

Pozadina svakog dobrog programskog jezika su njegovi okviri tj. njegove biblioteke. Python ima odlične i brojne biblioteke kao što su NumPy, Pandas, Keras, SciKit-Learn, Tensorflow, PyTorch itd. Njegove biblioteke imaju skoro sva rješenja za osnovne probleme.

3. 1. 2 Umjetna inteligencija i cloud servisi

Dvije najkorisnije stvari u tehnološkom svijetu današnjice koriste Python u pozadini. Python se pokazao odličnim jezikom kako za znanstvena istraživanja vezana uz duboko učenje tako i za razvoj efikasnih servisa u „oblacima“ *(eng. cloud service)* .

3. 1. 3 Automatizacija

U današnje vrijeme automatizacija je postala visoki prioritet tehnološkog svijeta, kao što su samovozni automobili, robotski usisavači i sl. što uvelike olakšava svakodnevni život. Python se smatra odličnim jezikom za automatizaciju zbog jednostavnosti izvedbe istih.

3. 2 Keras

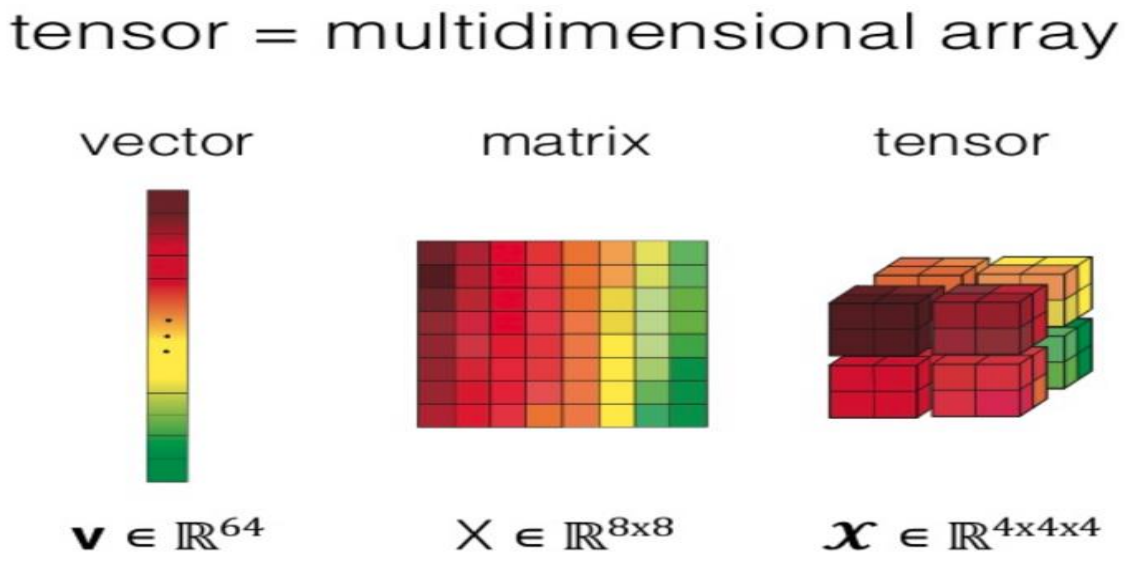
Keras je Pythonova biblioteka za neuronske mreže. Ova biblioteka je zaslužna za izuzetnu intuitivnost i jednostavnost manipulacije podataka te stvaranja modela i time omogućila velikom broju korisnika ulaz u svijet umjetne inteligencije. Inspiriran je Tensorflow-om te je zasnovan na slojevima. Keras efikasno i brzo odrađuje mrežna računanja uz funkcije visokog nivoa i daje im mogućnost širenja s potpuno prilagođenim mogućnostima. Dizajniran je da riješi problem paralelizma neuronskih mreža. Keras podržava treniranje ne samo na konvolucijskim neuronskim mrežama nego i na rekurentnim neuronskim mrežama.

3. 3 Tensorflow

Tensorflow je API dubokog učenja za Python napravljen od strane Google-a. Tensorflow se zasniva na četiri osnovna matematička pojma:

1. Skalar je tenzor koji sadrži samo jedan broj tj. tenzor s 0 dimenzija.
2. Vektor predstavlja tenzor s jednom dimenzijom, što znači da sadrži samo jednu os.
3. Matrice su nizovi vektora ili dvodimenzionalni tenor.
4. Nizovi su matrice s više od dvije dimenzije.

Tenzor je objekt koji opisuje linearnu povezanost između skalara, vektora i ostalih elemenata.



Slika 1. Prikaz objekata tenzora

Izvor: https://www.alibabacloud.com/blog/594604

Prednost Tensorflow-a je ta što za programsko računanje uz korištenje centralne procesorske jedinice nudi mogućnost korištenja grafičko procesorske jedinice. Tensorflow ne izvršava operacije na tradicionalan način, svaka matematička operacija koju računalo treba izvršiti predstavljena je kroz graf toka podataka. Grafovi se pripremaju s čvorovima koji se izvršavaju po grupama. Svaki čvor predstavlja jednu vrstu matematičke operacije (zbrajanje, dijeljenje, oduzimanje itd.).

Ključne odlike svakog tenzora su:

1. Broj osi ili broj činova, u Python modulu Numpy broj osi označava se s ndim.
2. Oblik tenzora označava se s skupom elemenata gdje jedan elemenata označava stupac a drugi elemenat označava red.
3. Vrsta podatka spremljenog u tenor.

3. 4 Ostale tehnologije

Uz gore navedene, Python ima veliki broj dodatnih biblioteka koje omogućavaju lakši rad s neuronskim mrežama te pomoćnim bibliotekama koje služe za vizualizaciju podataka i importiranja raznih tipova podatka u format razumljiv Pythonu.

3. 4. 1 Scikit-learn

Scikit-learn je Python modul koji nudi mogućnost implementacije prvoklasnih algoritama za nadzirano učenje uz neku osrednju količinu podataka te se može koristiti i za nenadzirano učenje. Izrađen je s naglaskom na:

1. Jednostavnost implementiranja
2. Optimiziranost
3. API učestalost

U pozadini koristi Numpy kao baznu podatkovnu strukturu za parametre i podatke. Posebno se ističe zbog upotrebe sučelja umjesto nasljeđa. Centralni objekt je prediktor *(eng. Estimator*) koji implementira *(eng. Fit)* metodu koja kao parametar prihvaća podatkovni niz i proizvoljno prihvaća niz oznaka vezanih za nadzirano učenje. Uz nju dolazi i metoda pretpostavi (*eng. Predict)* koja za ulazni podatak uzima validacijske podatke i vraća pretpostavljene oznake za odgovarajuće podatke.

3. 4. 2 Pandas

Pandas se očituje brzinom, fleksibilnošću i jednostavnošću korištenja. Otvorenog je koda što znači da svi mogu pridonijeti ovoj biblioteci. Koristi se za proučavanje i manipulaciju podataka. Svoj rad zasniva na još jednoj biblioteci zvanoj Numpy te je kompatibilna sa raznim modulima unutar Python ekosistema, tipično je unaprijed uključena u svaku distribuciju Pythona.

Neke od mogućnosti koje pruža Pandas:

1. Čišćenje podataka
2. Normalizacija podataka
3. Statistička analiza
4. Vizualizacija podataka
5. Učitavanje i spremanje podataka

3. 4. 2 Numpy

Bazična podatkovna struktura koja se koristi za podatke i modele. Naziv dolazi od brojčani Python *(eng. Numerical Python)*, a napisan je većinski u programskom jeziku C. Ulazni podatci su predstavljeni kao Numpy nizovi te se zbog toga besprijekorno uklapaju s ostalim znanstvenim bibliotekama Pythona. U sebi sadrži sveobuhvatne matematičke funkcije, generatore slučajnih brojeva te linearnu algebru.

3. 4. 3 Swift

Swift je programski jezik osnovan od strane Apple-a 2014. godine kao nadogradnja na tad već postojeći Objective - C. Koristi se za razvijanje iOS, macOS, watchOS te tvOS aplikacija. Swift kombinira značajke drugih programskih jezika kao što su Python, C te Objective-C te ga to čini svestranim jezikom. Zbog svoje brzine i jednostavnosti Swift je postao jedan od najpopularnijih jezika za izradu mobilnih aplikacija u Apple-ovom ekosistemu. Za potrebe ovog rada korištena je SwiftUI biblioteka, koja za razliku od klasičnog funkcionalnog programiranja, uzima pristup deklarativnog programiranja.

4 RAZVOJNO OKRUŽENJE

U ovom poglavlju opisani su alati kroz koje je odrađen praktični dio završnog rada uz naglasak na Anacondu. Za pisanje programskog koga korišten je Jupyter Notebook zbog lakše preglednosti i jednostavnijeg podešavanja radne okoline.

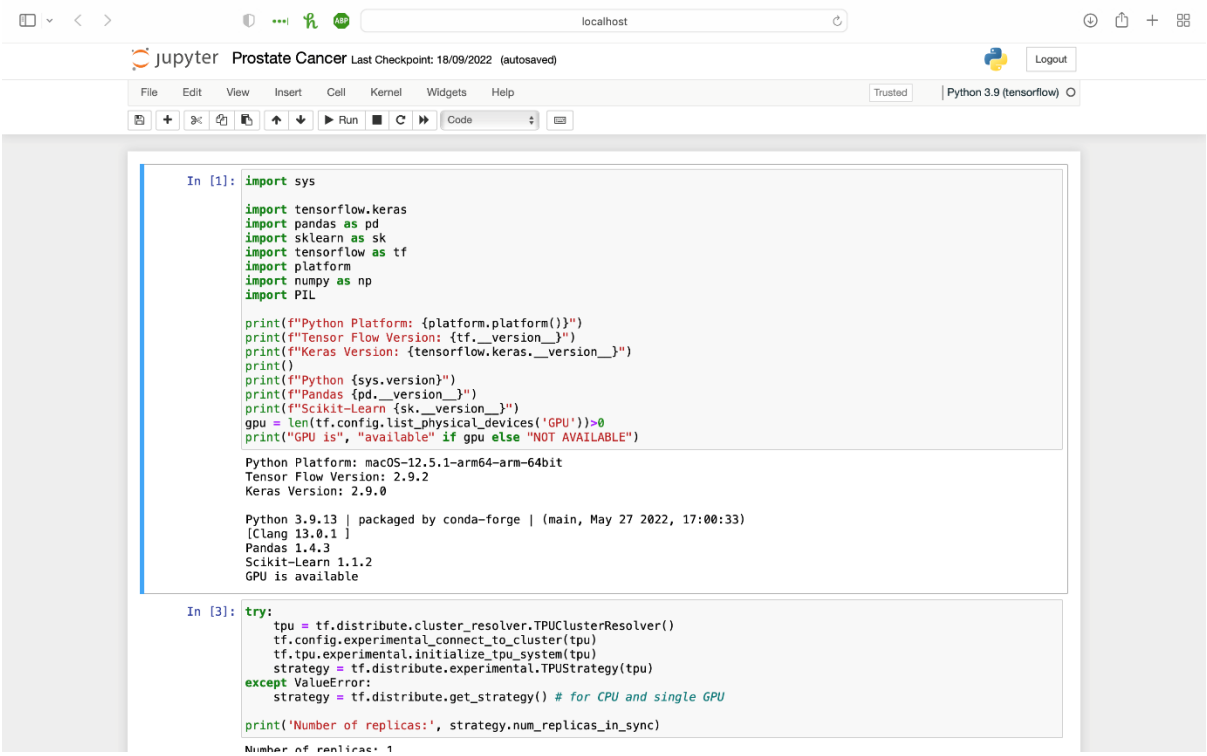
4. 1 Anaconda

Anaconda spada pod najpopularnija razvojna okruženja za rješavanje znanstvenih problema. Nudi jednostavno upravljanje integriranim aplikacijama, bibliotekama i razvojnim okruženjima bez upotrebe naredbenog retka. Podržava Windows, macOS i Linux. Napravljen je za Python ali podržava i ostale jezike kao što su: C, C++, FORTRAN, Java i mnoge druge. Anaconda podržava automatizaciju te konstantnu integraciju kroz korištenje integracijskih sistema kao što su Travis CI i AppVeyor koji omogućavaju automatsko testiranje koda. Anaconda nudi mogućnost između dvije verzije Pythona, Python 2 i Python 3.

4. 2 Jupyter Notebook

Jupyter Notebook sastoji se od takozvanih dokumenata bilježnice koja u sebi sadrži Python kod i elemente obogaćenog teksta E2.

Radi na način klijent - server gdje se dokumentima pristupa preko mrežnog preglednika, server se radi lokalno te za pokretanje Jupyter Notebook nije potrebna internetska veza. U pozadini se vrti jezgra (*eng. Kernel)* koja izvršava kod sadržan u blokovima. Python jezgra je zaslužna za izvršavanje programskog jezika Python. Kernel se otvara u pozadini skupa s Jupyter Notebook-om. Sve dok je server aktivan podatci se spremaju u radnu memoriju (*eng. RAM)* i brišu se tek nakon što se server ugasi te zbog toga korištenje Notebook-a može znatno usporiti rad računala.

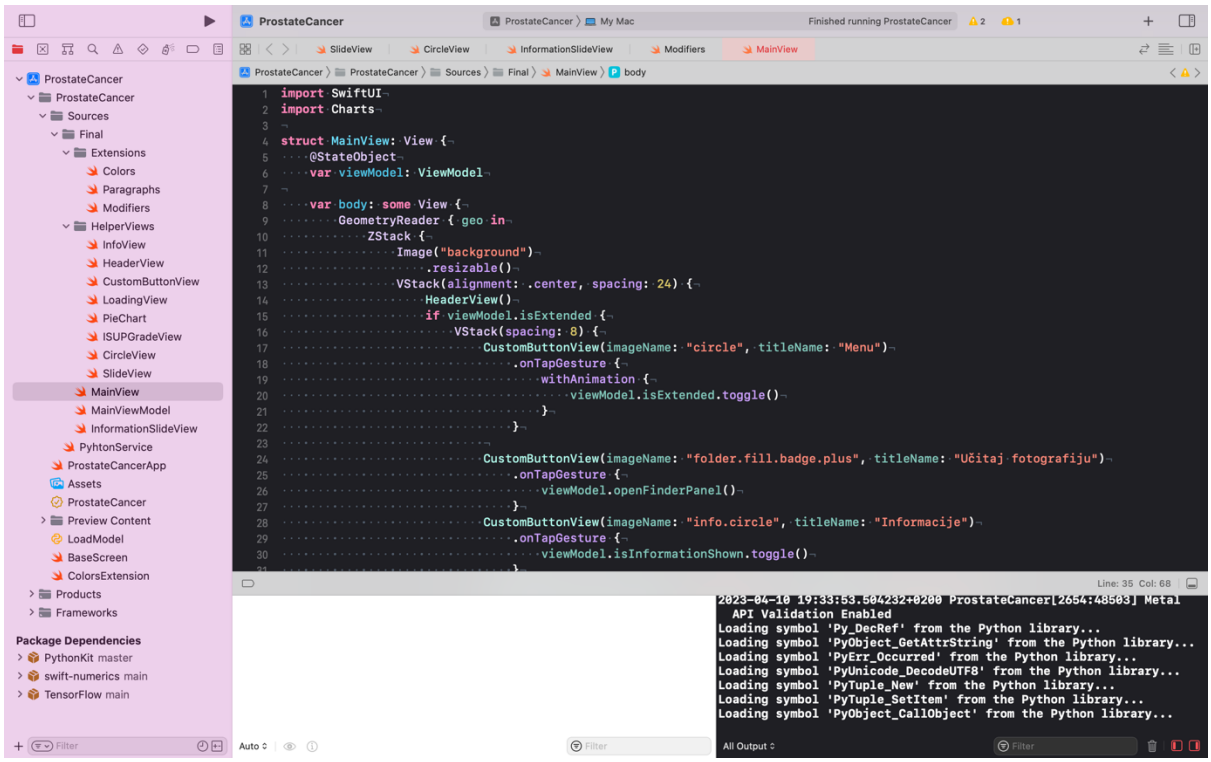


Slika 2. Jupyter Notebook

[Izvor: Autor]

4. 3 Xcode

Xcode predstavlja sveobuhvatnu integriranu razvojnu okolinu razvijenu od strane Apple Inc. za macOS operativne sisteme. Primarno se koristi za razvijanje aplikacija za raznolike Apple-ove platforme kao što su npr. macOS, iOS, watchOS i tvOS. Xcode pruža raznolike alate: za pisanje programskog koda te za potrebe debugiranja i testiranja. Sam po sebi Xcode ima integriranu kontrolu verzija (*eng. Source Controll)*, automatske testove te mogućnost distribucije aplikacija.



Slika 3 Prikaz Xcode IDE

[Izvor: Autor]

5. RAK PROSTATE

Rak prostate javlja se u prosjeku kod svakog osmog muškarca. Prostata je žlijezda orašastog oblika koja proizvodi sjemenu tekućinu. Jedan je od najučestalijih rakova kod muške populacije, uobičajeno sporo raste, vezana je za prostatu i vrlo često je bezazlen, ali neki tipovi su agresivni te zahtijevaju medicinsku asistenciju.

5. 1 Otkrivanje i testiranje

Kod otkrivanja raka prostate problematično je to što u ranim i srednjim fazama ne iskazuje nikakve jasno raspoznatljive simptome te se zbog toga jako često pronađe prekasno. Ako se ne tretira na vrijeme putem krvnih žila može se proširiti na ostale organe. Rak prostate otkriva se na dva načina:

1. Digitalni rektalni pregled

2. Test prostata-specifičnog antigena (PSA)

Ako ovi testovi ukažu na mogućnost postojanja neispravnosti kod prostate testiranje se dalje vrši ili ultrazvukom ili se uzima primjerak prostate biopsijom.

5. 2 Razina agresivnosti

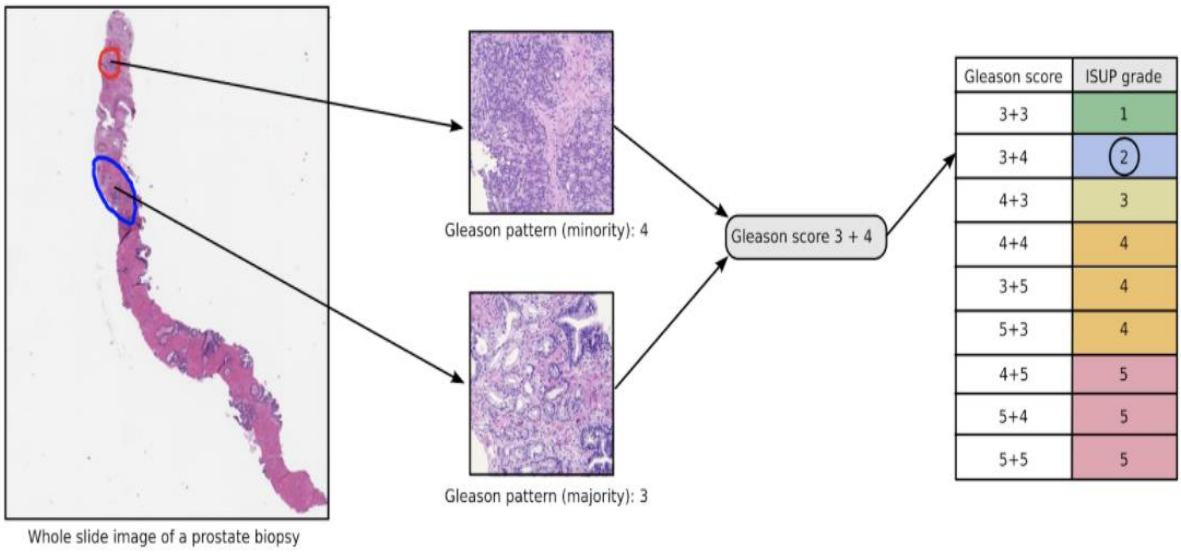
Nakon biopsije tkiva prostate sljedeći korak je vrednovanje agresivnosti ćelija raka. U medicini postoje dva standarda koji se koriste za opisivanje agresivnosti raka:

1. Gleason ljestvica: sastoji se od dva broja i označava se s arapskim brojevima od 2 do 10 gdje 10 označava najagresivniji.

2. ISUP skala: nastala je od sažimanja Gleason ljestvice na osnovu sljedećih pravila:

- Gleason vrijednost 6 = ISUP skala 1
- Gleason vrijednost 7 (3+4) = ISUP skala 2
- Gleason vrijednost 7 (4+3) = ISUP skala 3
- Gleason vrijednost 8 = ISUP skala 4
- Gleason vrijednost 9-10 = ISUP skala 5

Za svrhe ovog istraživanja koristit ćemo ISUP skalu radi jednostavnosti sažimanja



Slika 4 ISUP Grade

Izvor: https://storage.googleapis.com/kaggle-media/competitions/PANDA/Screen%20Shot%202020-04-08%20at%202.03.53%20PM.png

6. UMJETNA INTELIGENCIJA

Umjetna inteligencija je grana znanosti koja pokušava demonstrirati strojnu inteligenciju. U početku pojam umjetne inteligencije je opisivao računalo koje ima mogućnost rješavanja zadataka za koje se mislilo da su vezani samo uz živa bića. Dvije glavne grane umjetne inteligencije koje će biti opisane u radu su duboko učenje i strojno učenje.

6. 1 Strojno učenje

Strojno učenje se definira kao proučavanje programskih sistema koji uče sami te se prilagođavaju na osnovu prijašnjeg rezultata bez da su eksplicitno isprogramirani. Neki od problema za koji se strojno učenje pokazalo izuzetno učinkovito :

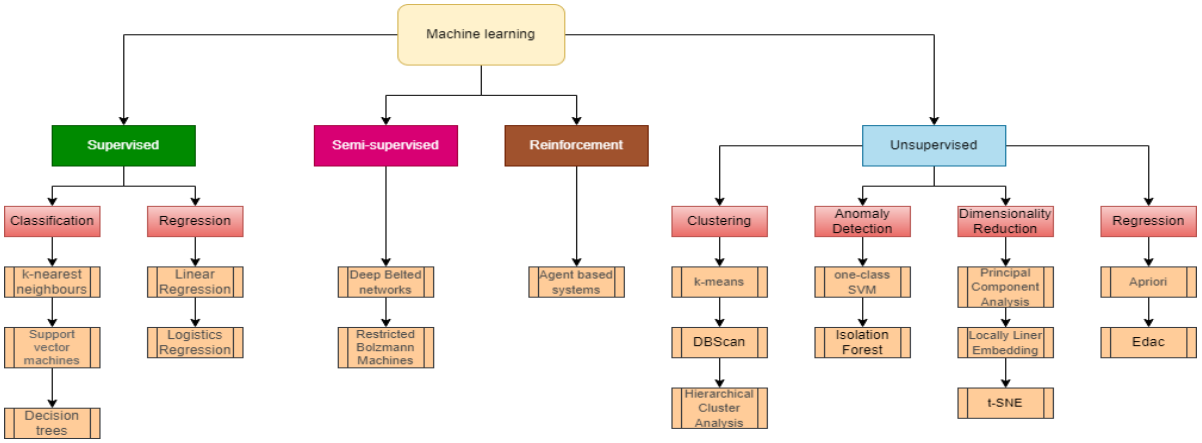
1. Pronalazak uzoraka i trendova u velikim podatkovnim skupovima

2. Kompleksni problemi koji nisu vraćali pozitivne rezultate kroz korištenje tradicionalnih metoda

3. Korištenje u dinamičkom prostoru gdje stroj može učiti iz svoje okoline

6. 1. 1 Podjela strojnog učenja

Postoji mnoštvo različitih vrsta strojnog učenja. Odabiranje prave vrste strojnog učenja za specifičan problem ovisi o mnoštvu faktora kao što su: količina podataka, koja se zahtijeva od modela te način treniranja modela ...



Slika 5: Grafički prikaz podjele strojnog učenja

[Izvor: Autor]

U ovom poglavlju opisane su dvije osnovne podjele strojnog učenja.

6.1.1.1. Nadzirano učenje (*engl. Supervised learning)*

Ova metoda radi na način da podatci za trening sadrže oznake koje označavaju željeni rezultat. Podatci se dijele u dva do tri skupa: skup za trening, testni skup te validacijski skup. Trening podatci se koriste da se poboljša model ili hipoteza, test podatci se koriste za ispravljanje parametara tokom učenja dok se validacijski podatci koriste za završnu procjenu točnosti modela. Hipoteza je matematička pretpostavka modela. Tokom procjene modela gleda se greška i na osnovu te greške se podešavaju ostali parametri modela. Postoje brojni algoritmi koji se koriste za treniranje modela koji će biti objašnjen u daljnjim poglavljima. Osnovna podjela nadziranog učenja je klasifikacija i regresija.

Odličan primjer klasifikacije je pretinac beskorisne elektronske pošte gdje algoritam sam nakon određenog vremena svrstava poštu u određene pretince.

Regresija se zasniva na predviđanju budućih vrijednosti na osnovu prošlih podataka. Uzmimo u obzir da se neki regresijski algoritmi mogu koristiti i za klasifikaciju i obrnuto.

6. 1. 1. 2. Nenadzirano učenje (*eng. Unsupervised learning)*

Nenadzirano učenje je potpuna suprotnost od nadziranog učenja kao što ime govori. U nenadziranom učenju trening podatci ne sadržavaju oznake koje označavaju željen rezultat nego je na računalu da ih označi. Tehnike koje se koriste za nenadzirano učenje uglavnom sadrže neku vrstu grupacije podatkovnih vrijednosti. Jedna vrlo bitna odlika nenadziranog učenja je njegova mogućnost pronalaska devijacija od norme. Primjer: pronalazak sumnjivih aktivnosti kreditnih kartica.

6. 2 Duboko učenje

Duboko učenje možemo opisati kao granu strojnog učenja koja je inspirirana strukturom i radom ljudskog mozga zvana umjetna neuronska mreža. Drugim riječima duboko učenje pokušava imitirati ponašanje moždanih funkcija. Algoritmi dubokog učenja su strukturirani na osnovu ljudskog živčanog sistema, gdje su svi neuroni povezani.

Modeli dubokog učenja su slojeviti te se prosječni model sastoji od najmanje 3 sloja. Svaki sloj prihvaća informacije od prijašnjeg i šalje ih sljedećem. Osnova dubokog učenja su linearna regresija i logistička regresija.

6. 2. 1 Linearna regresija

Linearna regresija je statistička metoda koja nam omogućava da sažmemo te proučavamo vezu između dvije količinske varijable. U slučaju dubokog učenja linearna regresija se koristi kako bi predvidjeli ponašanje kvantitativne varijable kroz stvaranje linearnih veza s jednom ili više nezavisnih osobina.

Kako bi rezultati linearne regresije bili valjani, podatci moraju ispunjavati određene uvjete kao što su:

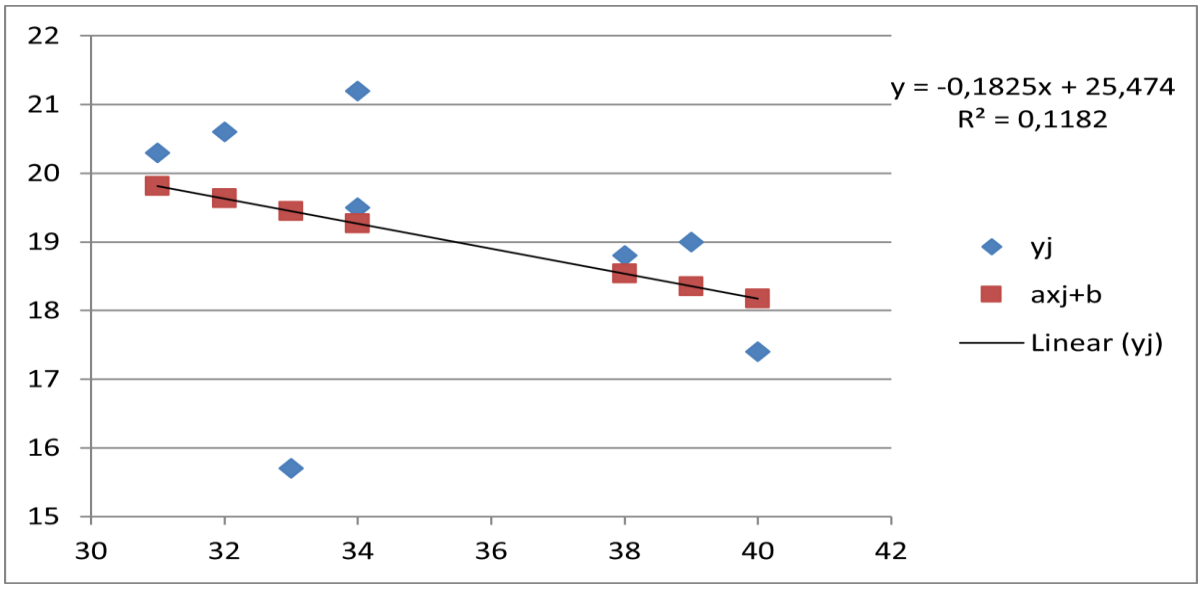
1. Svaka značajka u podatcima mora biti normalno distribuirana.

2. Multikolinearnost mora biti približna 0.

3. Aritmetička sredina razlike varijable y koju se promatra i pretpostavljene varijable y je 0.

4. Međusobna povezanost mora biti bliska 0.

Dvije glavne kategorije linearne regresije: Jednostavna linearna regresija i višestruka linearna regresija.



Slika 6. Linearna Regresija

[Izvor Autor]

6. 2. 2 Logistička regresija

Logistička regresija je inačica linearne regresije gdje je povratna varijabla dihotomna, tj. poprima vrijednosti 1 ili 0. Ideja iza logističke regresije je da pronađe vezu između značajki i vjerojatnosti određenog ishoda. Ovakvi tipovi problema često su zvani binomna logistička regresija gdje povratna informacija može imati samo dvije varijable, 1 ili 0, istina ili laž, da ili ne.

Kod binarne klasifikacije, neka je 'x' neka značajka i 'y' neka je rezultat koji može biti 1 ili 0. Vjerojatnost da je 'y' tj. rezultat jednaka jedan može se prikazati na sljedeći način:

$$P(y = 1 \,|x)$$

Ako je pretpostavka temeljena na linearnoj regresiji onda se izražava pomoću:

$$p(X) = \beta_0 + \beta_1 X$$

Logistička regresija može se izraziti kao :

$$\log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X$$

Lijeva strana jednadžbe predstavlja logit funkciju dok desna predstavlja šansu. Šansa predstavlja omjer između vjerojatnosti uspjeha i vjerojatnosti neuspjeha.

Izvedbu logističkog modela provjeravamo kroz devijaciju srednjih kvadratnih vrijednosti te još možemo vizualno prikazati izvedbu modela kroz matricu zabune.

	P' (Pretpostavljena)	n' (Pretpostavljena)
P	Pozitivna Istina	Negativna Laž
n	Pozitivna Laž	Negativna Istina

Tablica 1. Matrica Zabune [1]

[Izvor: Autor]

Točnost modela se računa :

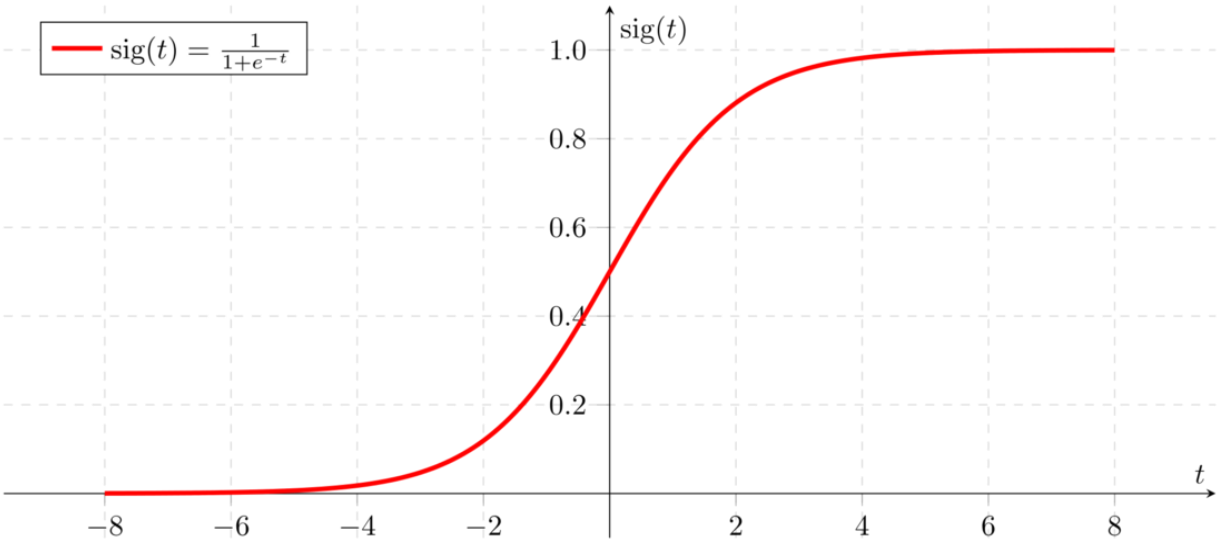
$$\frac{Pozitivna\ istina + Negativna\ Laž}{Pozitivna\ istina + Negativna\ istina + Positivna\ laž\ + Negativna\ laž}$$

6. 2. 3 Aktivacijske funkcije

Aktivacijska funkcija dodaje se u umjetnu neuronsku mrežu radi lakšeg pronalaska kompleksnih uzoraka u podatcima. Kod usporedbe s ljudskim živčanim stanicama, aktivacijska funkcija stoji na kraju neurona i odlučuje što se šalje sljedećem neuronu. Dok kod umjetne neuronske mreže aktivacijska funkcija prima izlaz od prošle stanice, oblikuje podatak u željenu formu koja odgovara sljedećem neuronu te ga šalje dalje. Glavna uloga aktivacijskih funkcija u neuronskim mrežama je unošenje nelinearnosti i stvaranje novih izlaznih vrijednosti.

Neke od vrste aktivacijskih funkcija:

1. Sigmoidna aktivacijska funkcija je poprilično zastupljena u dubokom učenju zbog olakšavanja povratnog razmnožavanja (*eng. Back Propagation)*



Slika 7. Sigmoidna Funkcija

[Izvor: Autor]

Formula glasi:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \dots (1)$$

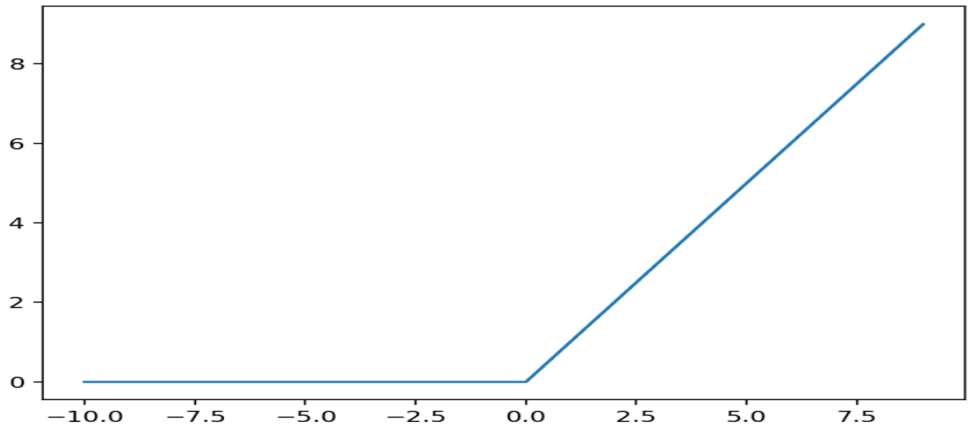
Glavni problem sigmoidne funkcije je zasićeni gradijent. Kako vrijednosti funkcije osciliraju između nula i jedan, postoji šansa da vrijednosti postanu konstante te gradijentni skup izgubi smisao jer nema promjena u vrijednostima.

2. Hiperbolička tangenta aktivacijska funkcija

Hiperbolička tangentna aktivacijska funkcija najprikladnija je za mreže gdje je ulaz veći od 1. Za posljedicu vrijednosti gradijenta su sve ili pozitivne ili negativne te isto kao sigmoidna funkcija susreće se s problemom zasićenog gradijenta.

3. RELU *(eng. Rectified Linear Unit Activation Function)*

Ispravljena funkcija aktiviranja linearne jedinice jedna je od najpopularnijih aktivacijskih funkcija. Za razliku od sigmoidne aktivacijske funkcije, ispravljena funkcija aktiviranja linearne jedinice nema problema s prezasićenosti gradijenta te zahtijeva manju računalnu moć.

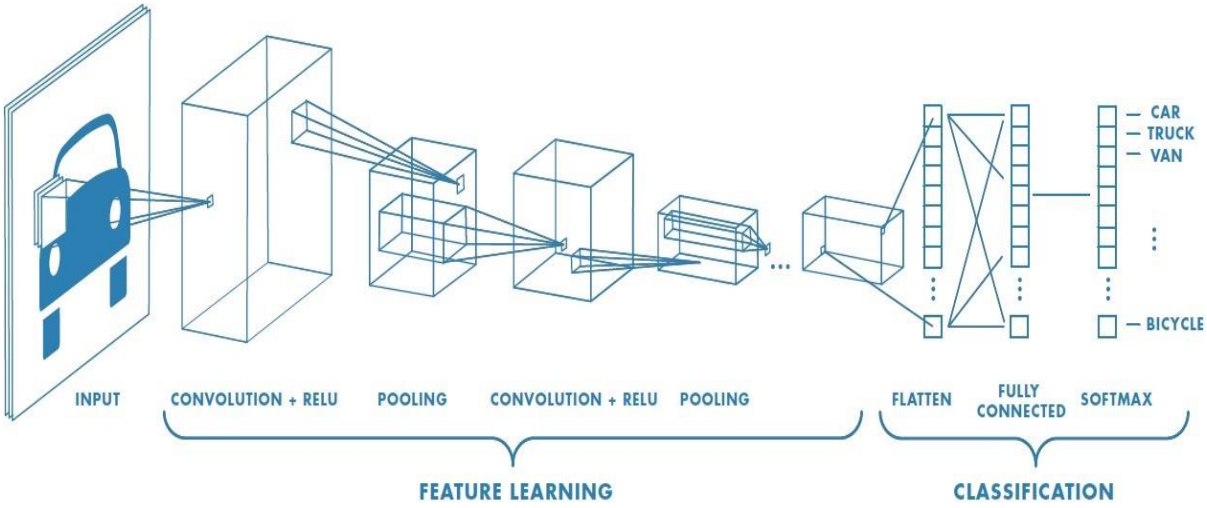


Slika 8. RELU

[Izvor: Autor]

6. 3 Konvolucijske neuronske mreže

Konvolucijske neuronske mreže pojavile su se 1998. kao pokušaj prepoznavanja poštanskih brojeva, ali tek 2012. su postale osnove dubokog učenja i klasificiranja podataka. Na globalnom natjecanju koje je zastupljeno od strane Image Net, Alex Net je uz pomoć konvolucijske neuronske mreže uspjela smanjiti grešku za 10% te otvorila nova vrata za polje umjetne inteligencije. Duboko učenje i konvolucijske neuronske mreže su usko povezane, sve vrste klasificiranja, raspoznavanja objekata se vrši pomoću konvolucijskih neuronskih mreža. U srži konvolucijske neuronske mreže kao ulazni parametar uzimaju sliku, procesiraju istu te klasificiraju u jednu od priloženih kategorija. Ulazni podatak tj. sliku računalo predstavi pomoću niza piksela koje ovise o razlučivosti slike i različitim dimenzijama slike. Kao osnovne podatke model uzima širinu, visinu i dimenziju slike. Svaka slika prolazi kroz serije konvolucijskih slojeva s određenim filterima (*eng. Kernels)*, slojevi sažimanja, potpuno povezani slojevi te na kraju se poziva Softmax funkcija.



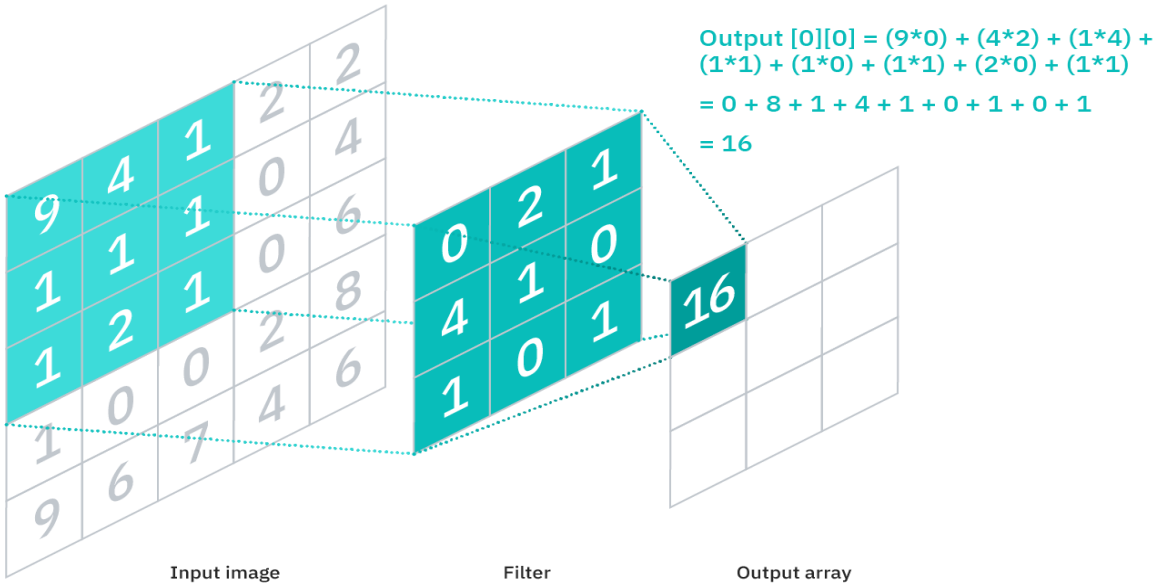
Slika 9. Neuronska mreža s mnoštvom konvolucijskih slojeva

*Izvor : https://miro.medium.com/max/1400/1*XbuW8WuRrAY5pC4t-9DZAQ.jpeg*

6. 3. 1 Konvolucijski sloj

Konvolucijski sloj stoji u srži konvolucijskih neuronski mreža gdje se većina matematičkih operacija izvršava. Zahtijeva par komponenti kao što su ulazni podatci, filter te kartu značajki. Konvolucijski sloj zadužen je za očuvanje veze između piksela te pronalazak značajki koji se izvršava pomoću filtera ili kernela.

- Matrica slike dimenzija (v, š, d)
- Filter ($f_n \times f_w \times d$)
- Izlaz volumen dimenzija$(h - f_h + 1) \times (w - f_w + 1) \times 1$



Slika 10. Prikaz umnoška matrice i filtera

] Izvor: https://1.cms.s81c.com/sites/default/files/2021-01-06/ICLH_Diagram_Batch_02_17A-ConvolutionalNeuralNetworks-WHITEBG.png

Nakon što slika prođe kroz konvolucijski sloj, ostaje nam matrica s puno manje parametara tj. slika se sažme u jednostavniju matricu koja omogućava ostalim slojevima jednostavniju manipulaciju. Ako se slika nn provuče kroz filter $k * k$, izlazna slika bit će dimenzija $(n - k + 1) \times (n - n + 1)$. Korištenjem raznih filtera omogućava izvršavanje raznih operacija kao što su prepoznavanje rubova, zamućenje dijelova slike te izoštravanje dijelova slike. Filter je dvodimenzionalni niz težinskih vrijednosti. Postoje varijacije u veličini filtera ali prosjek je matrica veličine 3x3.

Filter se nanosi na određeni dio slike, kao rezultat dobije se broj izračunat uz ulazni piksel i filter. Rezultat se dalje stavlja u novu matricu te se filter pomiče za jedan korak, a proces se

ponavlja dok filtar ne prođe kroz cijelu sliku. Važno je zapamtiti da se težinske vrijednosti filtera ne mijenjaju. Postoje tri parametra koji mogu utjecati na rezultat konvolucijskog sloja:

1. Broj filtera utječe na dubinu izlaznog podatka.
2. Koraci tj. udaljenost ili broj piksela kojim se filter kreće po ulaznoj matrici.
3. Nulto punjenje se koristi kad filter ne odgovara ulaznoj slici. Postavi sve elemente koje spadaju izvan ulazne matrice na nulu.

Konvolucijski sloj koristi ispravljenu linearnu aktivacijsku funkciju za pronalazak značajki. Konvolucijska neuronska mreža može imati više konvolucijskih slojeva koji su organizirani hijerarhijski. Na kraju konvolucijski sloj pretvara slikovni podataka u numerički podataka.

6. 3. 2 Sloj sažimanja

Sloj sažimanja, znan kao sloj smanjivanja uzoraka, provodi sažimanje dimenzija ulaznog podatka te sažima broj parametara kod ulaznog podatka. Radi na sličan način kao konvolucijski sloj, tokom sažimanja filter prođe kroz cijelu sliku, jedina razlika je što filter kod sloja sažimanja za razliku od konvolucijskog sloja ne sadrži težinske vrijednosti. Umjesto težinskih vrijednosti filter nanosi funkciju agregacije prijemnim poljima, na taj način puni novu matricu. Postoje dvije osnovne vrste sažimanja:

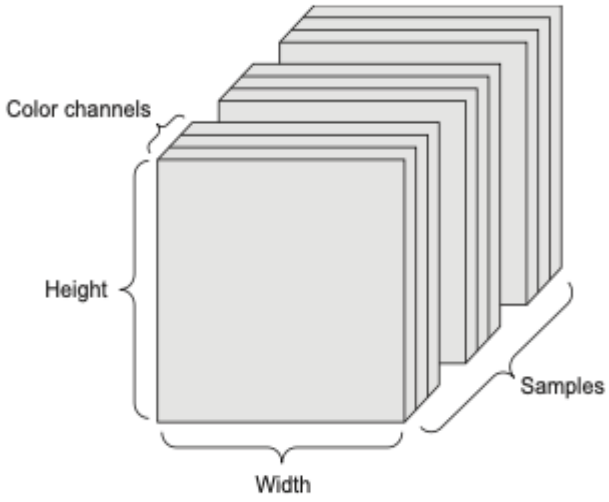
1. Maksimalno sažimanje: Kako filter prolazi putanjom kroz ulaznu sliku, odabire najveću vrijednost te ju šalje dalje.
2. Prosječno sažimanje: Kako se filter miče kroz ulazni podataka uzima prosjek prijemnog polja i šalje ga kao izlazni podatak.

6. 3. 3 Potpuno povezani sloj

Potpuno povezani sloj nam opisuje sam sebe. U prijašnjim slojevima vrijednosti ulaznih piksela i izlaznih nisu direktno povezane. Što znači da je svaki izlazni čvor trenutnog povezan s čvorom prijašnjeg sloja. Uloga potpuno povezanog sloja je da odradi razvrstavanje s obzirom na odlike izvučene kroz prijašnje slojeve. Zadnja bitna odlika potpuno povezanog sloja je razlika u aktivacijskim funkcijama naspram konvolucijskog sloja te sloja sažimanja je korištenje softmax aktivacijske funkcije umjesto ispravljene aktivacijske funkcije. Izlazna vrijednost predstavlja vjerojatnost da određena slika pripada određenoj klasi, označena vrijednostima između 0 i 1.

6. 4 Reprezentacija slika u dubokom učenju

Klasične slike sadrže tri dimenzije: visinu, širinu i dubinu boja. Slike se po konvenciji prikazuju kao 3D tenzor.



Slika 11. 4D reprezentacija slike kroz tenzor

Izvor: https://www.kaggle.com/code/paulrohan2020/scaler-vector-tensor-basics-for-neural-network

Dva osnovna načina zapisivanja slikovnog podatka u tenzor su posljednji kanal *eng.(channels-last)* korišten od strane Tensorflow-a i prvi kanal *eng.(channels-first)* korišten od stane Theano.

6. 5 Hiperparametri mreže

Hiperparametri mreže predstavljaju varijable koje definiraju strukturu mreže te način na koji mreža uči. Hiperparametri se postavljaju prije početka treniranja i prije postavljanja težinskih vrijednosti i pristranosti mreže. Dijelimo ih na dvije vrste parametara. Parametri vezani za strukturu mreže kao što su broj skrivenih slojeva tj. broj slojeva između ulaznog i izlaznog sloja. U nekim slučajevima isključivanje nasumično odabranih neurona radi poboljšanja točnosti tj. smanjivanja mogućnosti preučenog modela te inicijalizacija težinske vrijednosti mreže. Zadnji parametar vezan za strukturu mreže ujedno jedan od najvažnijih je aktivacijska funkcija. Druga vrsta hiperparametara vezana je za algoritme učenja. Prvi i osnovni parametar predstavlja stopa učenja. Stopa učenja jedan je od najutjecajnijih parametara koji se dodjeljuju algoritmima. Ako je stopa učenja niska proces učenja se usporava, visoka stopa učenja ubrzava učenje, ali istovremeno smanjuje mogućnost konvergencije. Broj epoha predstavlja broj koji odlučuje koliko puta mreža prolazi kroz podatke. Nakon postavljanja epoha, postavlja se broj serija tj. broj uzoraka nakon kojeg se parametri mreže ažuriraju.

6. 6 Učenje s prijenosom znanja

Učenje s prijenosom znanja je tehnika koja uzima već istreniran model za određeni problem i implementira ga za potrebe trenutnog problema. Ova tehnika oslanja se na iskustvo koje je model stekao na jednom problemu te da to isto iskustvo prenese tj. omogući lakšu generalizaciju na trenutnom problemu. Tehnika učenja s prijenosom znanja najučestalija je na područjima računalnog vida te procesiranje prirodnog jezika. Ova tehnika ne spada pod tehnike strojnog učenja nego više opisuje metodologiju dizajna. Pretežno se koristi za smanjivanje vremena potrebnog za učenje te smanjuje potrebu za ogromnim količinama podataka. Postoje tri pristupa kod korištenja učenja s prijenosom znanja:

1. Treniranje modela za prijenos znanja
2. Korištenje već istreniranih modela
3. Izdvajanje značajki

6. 6. 1 Primjeri poznatih arhitektura

U vrijeme kad se umjetna inteligencija koristi na svakom koraku omogućuje nam veliki broj već istreniranih modela za korištenje. Za potrebe ovoga rada opisati ću dva najzastupljenija modela za proces računalnog vida, a to su VGG16 i Alex Net.

6. 6. 1. 1 VGG16

Osnivači VGG16 su profesori na Visual Geometry Group, University of Oxford. Ime proizlazi iz njihovog polja znanosti. VGG16 pripada skupini konvolucijskih slojeva u dubokom učenju te se danas smatra jednim od najboljih modela programskog vida. Sastoji se od veoma malih konvolucijskih filtera (3x3) što je pokazalo značajno poboljšanje u usporedbi s prijašnjim konvolucijskim filterima. Definira se kao algoritam za klasificiranje i raspoznavanje koji može 1000 slika razvrstati u 1000 različitih kategorija s točnošću od 92.7%. Broj 16 predstavlja broj slojeva koje sadrže težinske vrijednosti, od njih šesnaest, trinaest ih pripada konvolucijskim slojevima, pet ih pripada sloju maksimalnog punjenja (*eng. Max Pooling)* i tri zadnja tri sloja pripadaju gustom sloju *(eng. Dense layer)*. Za ulazni podatak prima tenzor veličine (224,224, 3). VGG16 se ističe od ostalih modela zbog:

1. Smanjenog broja hiperparametara
2. Veličina koraka je uvijek 1
3. Veličina punjenja je uvijek ista

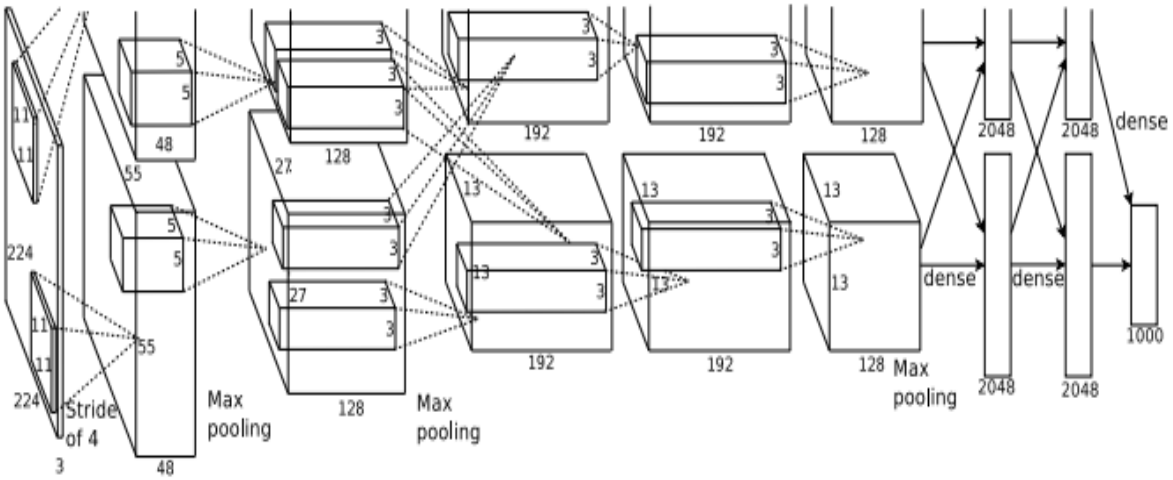
```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_1 (InputLayer)         [(None, 1536, 128, 3)]    0

 block1_conv1 (Conv2D)        (None, 1536, 128, 64)     1792

 block1_conv2 (Conv2D)        (None, 1536, 128, 64)     36928

 block1_pool (MaxPooling2D)   (None, 768, 64, 64)       0

 block2_conv1 (Conv2D)        (None, 768, 64, 128)      73856

 block2_conv2 (Conv2D)        (None, 768, 64, 128)      147584

 block2_pool (MaxPooling2D)   (None, 384, 32, 128)      0

 block3_conv1 (Conv2D)        (None, 384, 32, 256)      295168

 block3_conv2 (Conv2D)        (None, 384, 32, 256)      590080

 block3_conv3 (Conv2D)        (None, 384, 32, 256)      590080

 block3_pool (MaxPooling2D)   (None, 192, 16, 256)      0

 block4_conv1 (Conv2D)        (None, 192, 16, 512)      1180160

 block4_conv2 (Conv2D)        (None, 192, 16, 512)      2359808

 block4_conv3 (Conv2D)        (None, 192, 16, 512)      2359808

 block4_pool (MaxPooling2D)   (None, 96, 8, 512)        0

 block5_conv1 (Conv2D)        (None, 96, 8, 512)        2359808

 block5_conv2 (Conv2D)        (None, 96, 8, 512)        2359808

 block5_conv3 (Conv2D)        (None, 96, 8, 512)        2359808

 block5_pool (MaxPooling2D)   (None, 48, 4, 512)        0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Slika 12. Prikaz slojeva VGG16 modela

[Izvor: Autor]

6. 6. 1. 2 AlexNet

AlexNet isto kao VGG16 proizlazi iz prestižnog Sveučilišta Toronto. Sastoji se od 8 slojeva, 5 slojeva su konvolucijski slojevi te 3 potpuno povezana sloja. Za aktivacijsku funkciju koristi se ispravljenom linearnom aktivacijskom funkcijom za razliku od tadašnjeg standarda tahn funkcije. Karakteristika koja ističe AlexNet od konkurencije je mogućnost korištenja više od jedne grafičke računalne jedinice *(eng. GPU)*, što smanjuje vrijeme potrebno za treniranje modela. Alex net koristi punjenje s preklapanje za razliku od ostalih modela.



Slika 13. Prikaz slojeva AlexNet Modela

Izvor https://iq.opengenus.org/content/images/2019/01/alexnet-1.png

7. PRAKTIČNI DIO

U praktičnom dijelu rada detaljno je razrađen postupak prikupljanja i obrade slikovnih podataka te proces učenja modela na danim podatcima. Slikovni podatci se obrađuju uz pomoć Tensorflow metoda te pretvaraju u oblik prihvatljiv modelu. Podatci za učenje se sastoje od skupa slika od kojih svaka sadrži jedinstveno ime te .csv *(emg. Comma separated values)* datoteke s podatcima o svim slikama te njihovim odlikama. Vizualno je pomoću grafova prikazan odnos podataka na osnovu ISUP skale. Prikazani su primjeri slika korišteni za treniranje modela u matrici veličina 10 x 10. Nakon što su podatci spremni za treniranje, sljedeći korak je definiranje arhitekture modela uz pomoć učenja s prijenosom znanja. U procesu definiranja arhitekture modela koriste se razne translacijske metode kako vertikalne tako i horizontalne koje su zaslužne za uvođenje nasumičnosti kod trening skupine. Iskorištena je tehnika učenja s prijenosom znanja kroz korištenje VGG16 modela s ImageNet težinskim vrijednostima te mu se dodaju dodatni slojevi kao što su: GlobalAveragePooling2D, Dense, BatchNormalization koji su zaslužni za prilagodbu modela određenomu problemu. Kao optimizator implementiran je RMSProp te za funkciju gubitka koristi se categorical_crossentropy čiji su standardni parametri za korištenje kod problema prepoznavanja slika. Uz korištenje osnovnih slojeva za kreiranje arhitekture modela korištene su određene metode pretprocesiranja slika zbog mogućnosti poboljšanja rezultata modela. Prije nego model može započeti proces treniranja definirana je stopa učenja. Na kraju iteracije učenja prikazani su rezultati naučenog modela.

7. 1 Prikupljanje podataka

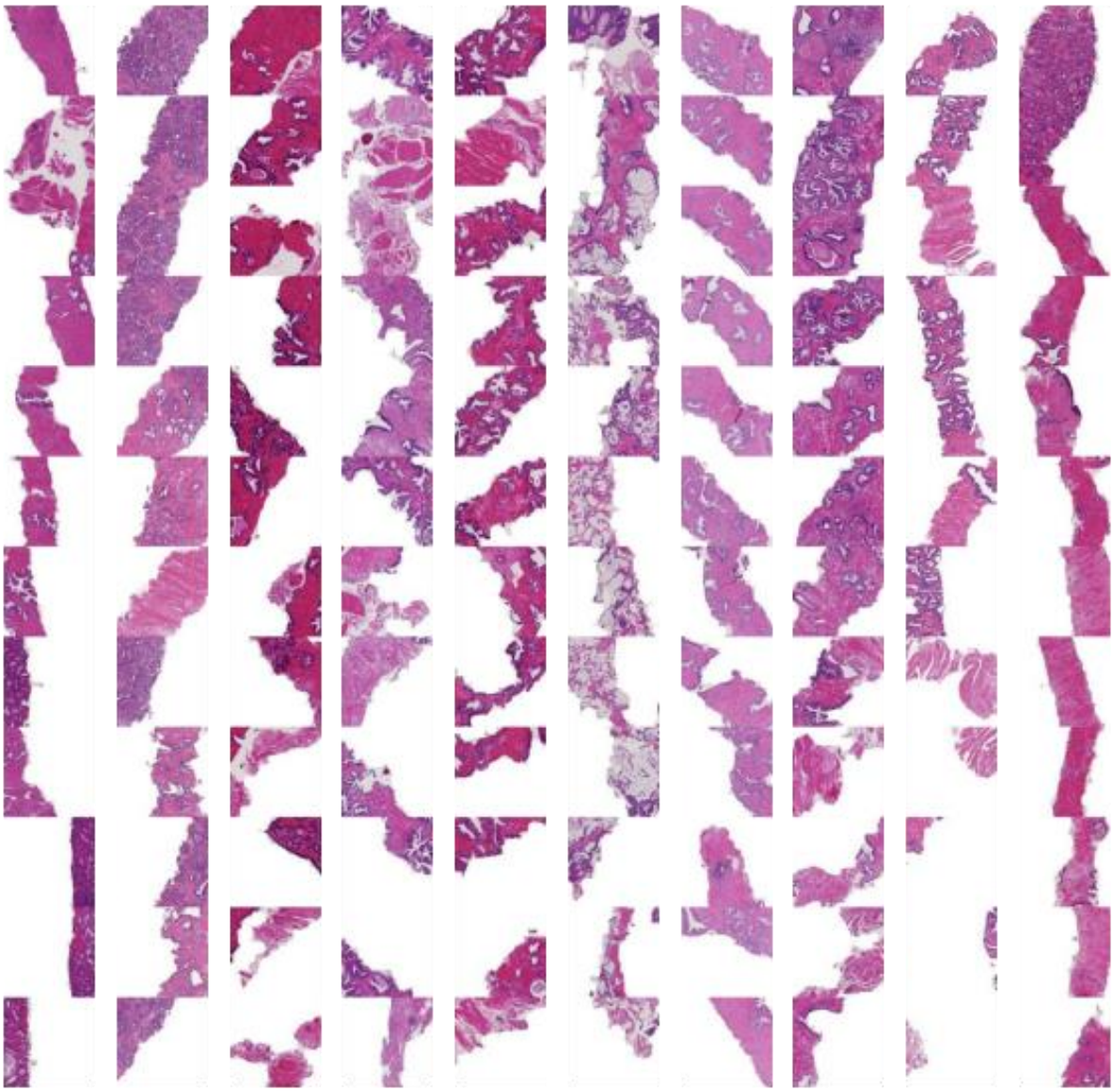
Razliku između dobre i loše procjene modela možemo prepoznati na osnovu ulaznih podataka. Slike korištene u svrhu završnog rada preuzete su s Kaggle Internet stranice te dolaze od Karolinskog instituta. Kaggle je web stranica posvećena svim vrstama umjetne inteligencije. Sadrži ogroman broj podatkovnih skupova i besplatna je. Podatkovni set sastoji se od 5456 slika, različitih dimenzija te u .jpg formatu. Informacije vezane za svaku od jedinstvenih slika spremljene su u formatu vrijednosti odvojene zarezom. U sljedećim koracima pripremat ćemo kako slikovne tako i tekstove podatke, cilj je olakšati modelu prolazak kroz cijeli podatkovni set.

	isup_grade	image_id
0	0	1925
1	1	1814
2	2	668
4	4	481
3	3	317
5	5	251

Slika 14. Broj slika po ISUP Grade

[Izvor: Autor]

Zbog jednostavnijeg učenja modela slike su izrezane u više manjih dijelova te posložene vertikalno. Podatci se dijele na tri skupa: skup za trening, skup za test i skup za validaciju.



Slika 15. Prikaz slika raka prostate

[Izvor: Autor]

7. 2 Obrada podataka

Obrada podataka smatra se jednim od najvažnijih koraka u procesu učenja modela. Način na koji se podatci prikažu modelu može znatno poboljšati ili otežati proces učenja. Prije nego je moguće obrađivati podatke potrebno ih je učitati u bilježnicu. Podatke koristimo na osnovu putanje do određenog dokumenta. Nakon podjele podataka ostajemo s 3 polja podataka u programskom jeziku Python:

- `k_train` – označava trening skupinu
- `k_test` – označava testnu skupinu
- `k_valid` – označava validacijsku skupinu

```
MAIN_DIR = '/Users/mateoperic/Documents/Zavrsni Rad/
archive'

TRAIN_IMG_DIR = os.path.join(MAIN_DIR,'all_images')

train_csv=pd.read_csv(os.path.join(

MAIN_DIR, 'data_train.csv'))

karloinska_csv = train_csv[train_csv['data_provider'] !=
'radboud']

k_train, k_test = train_test_split(

karloinska_csv,

test_size=0.2,

random_state = SEED)


k_train, k_valid = train_test_split(

k_train,

test_size= 0.25,

random_state = SEED)
```

Nakon što su podatci podijeljeni u željene skupine potrebno je urediti .csv dokument zbog jednostavnosti prolaska kroz podatkovni set.

data_train				
	image_id	data_provider	isup_grade	gleason_score
0	0005f7aaab2800f6170c399693a96917	karolinska	0	0+0
1	000920ad0b612851f8e01bcc880d9b3d	karolinska	0	0+0
2	0018ae58b01bdadc8e347995b69f99aa	radboud	4	4+4
3	001c62abd11fa4b57bf7a6c603a11bb9	karolinska	4	4+4
4	001d865e65ef5d2579c190a0e0350d8f	karolinska	0	0+0
5	002a4db09dad406c85505a00fb6f6144	karolinska	0	0+0
6	003046e27c8ead3e3db155780dc5498e	karolinska	1	3+3
7	0032bfa835ce0f43a92ae0bbab6871cb	karolinska	1	3+3
8	003a91841da04a5a31f808fb5c21538a	karolinska	1	3+3

Slika 16. Prikaz .csv datoteke prije obrade podataka

[Izvor Autor]

Kao što se vidi iz prikazanog, podatci u tabeli ne sadrže putanju određene slike. U sljedećem koraku povezat ćemo ISUP skalu i putanju datoteke.

```
k_train['isup_grade'] = k_train['isup_grade'].apply(str)

k_train['file'] = TRAIN_IMG_DIR + '/_' + k_train['isup_grade']
+ '_' + k_train['image_id'] + '.jpg'


k_test['isup_grade'] = k_test['isup_grade'].apply(str)

k_test['file'] = TRAIN_IMG_DIR + '/_' + k_test['isup_grade'] +
'_' + k_test['image_id'] + '.jpg'
```

Prvi korak kod uređivanja tekstualnih podataka je pretvoriti sve vrijednosti koje nisu tekstualne u tekstualne podatke. Nakon što imamo sve podatke u istom formatu dodajemo novo polje 'file' koje sadrži putanju do direktorija svih slika, ISUP vrijednost i jedinstven naziv slike.

	Unnamed: 0	image_id	data_provider	isup_grade	gleason_score	file
5424	5472	86e1e1f8d6a8d630b1577b3a5f3d65a3	karolinska	2	3+4	/Users/mateoperic/Documents/Zavrsni Rad/Untitl...
4534	4576	714a9564b61a4daee6c8a39a61e04500	karolinska	1	3+3	/Users/mateoperic/Documents/Zavrsni Rad/Untitl...
3848	3880	60c90d72afb6a820a0a539bc97a144fb	karolinska	1	3+3	/Users/mateoperic/Documents/Zavrsni Rad/Untitl...
8179	8255	c756f85302c24df011743d63b5282d58	karolinska	0	0+0	/Users/mateoperic/Documents/Zavrsni Rad/Untitl...
1167	1178	1dfa9e0e91f817fbd98ae93fc20d6255	karolinska	5	5+5	/Users/mateoperic/Documents/Zavrsni Rad/Untitl...

Slika 17. Primjer .csv datoteke nakon dodavanja 'file' stupca

[Izvor Autor]

Nakon što je .csv datoteka spremna za rad potrebno je definirati metodu koja slikovne podatke dekodira u trodimenzionalnu matricu. Metoda `decode_img(img)` za ulazni parametar uzima slikovni podatak te ga prvobitno pretvara u `uInt8` vrijednost, nakon prvobitne pretvorbe potrebno je tip pretvoriti u `float` s vrijednostima između 0 i 1. Zadnji korak je promjena razlučivosti slike u željeni format.

```
IMG_DIM = (1536, 128)

def decode_img(img):

    img = tf.image.decode_jpeg(img, channels=3)

    img = tf.image.convert_image_dtype(img, tf.float32)

    return tf.image.resize(img, IMG_DIM)
```

Metoda vraća podatak tipa `float32` te ga predstavlja kao matricu veličine (1536 x 128). Nakon definiranja metode za dekodiranje slikovnih podataka potrebno je deklarirati metodu `get_item(file_path)` koja na osnovu putanje izvedene iz .csv dokumenta vraća dekodiranu sliku te ISUP vrijednost podatka. Definirana varijabla `image` koja predstavlja sliku na osnovu putanje `file_path` se dekodira uz korištenje prijašnje definirane funkcije `decode_img(img)`. Zadnji korak je definiranje ISUP vrijednosti vezane za podataka na putanji `file_path`. Ulazni parametar `file_path` sastoji se od:

- Putanje do glavnog direktorija
- ISUP vrijednosti
- Jedinstvenog naziva

Podatci su odvojeni '_' zbog jednostavnijeg raspoznavanje informacija. Varijablu `label` definiramo tako što ulazni parametar rastavimo na njegove dijelove korištenjem `tf.string.split`, nakon što smo podijelili niz na više manjih nizova potrebno je dohvatiti posljednje dvije vrijednosti koje označavaju ISUP vrijednost i jedinstven naziv odabranog podatka. Dohvaćenu ISUP vrijednost potrebno je pretvoriti iz `string` u `int32` korištenjem `tf.cast` metode. Varijabla `CLASSES_NUM` označava broj svih mogućih ISUP vrijednosti.

```
CLASSES_NUM = 6

def get_item(file_path):

    image = tf.io.read_file(file_path)

    image = decode_img(image)

    label = tf.strings.split(file_path, '_')

    label = tf.strings.to_number(label[-2])

    label = tf.cast(label, tf.int32)

    return image, tf.one_hot(label, CLASSES_NUM)
```

Metoda vraća dekodiranu sliku i enkodirani tenzor.

7. 3 Definiranje seta za učenje

Kako bi definirali podatkovni set za učenje potrebno je povezati prijašnje metode. Informacije iz .csv dokumenta pod poljem 'file' potrebno je pretvoriti u format koji za svaki podataka u matrici stvara objekt. Korištenjem `tf.data.Dataset.from_tensor_slices` metode koja za ulazni parametar uzima bilo kakav niz podataka. Nakon što su podatci spremljeni u zasebne objekte potrebno je proći kroz sve generirane objekte te ih kao ulazni parametar dati već definiranoj funkciji `get_item(file_path`. Korištenjem funkcije `map()` primjenjuje funkciju `get_item` na svaki element podatkovnog seta, uz paralelizaciju od `num_parallel_calls` bez potrebe korištenja `for` petlje. Prije nego podatkovni set bude spreman za treniranje potrebno je dodati element nasumičnosti korištenjem `shuffle` metode. Nakon što su podatci nasumično posloženi potrebno je odrediti broj elemenata po iteraciji učenja korištenjem `batch` metode.

```
AUTOTUNE = tf.data.experimental.AUTOTUNE

BATCH_SIZE = 32 * strategy.num_replicas_in_sync

def get_dataset(df):

    ds = tf.data.Dataset.from_tensor_slices(df['file'].values)

    ds = ds.map(get_item, num_parallel_calls = AUTOTUNE)

    ds = ds.shuffle(buffer_size=1000)

    ds = ds.batch(BATCH_SIZE)

    ds = ds.prefetch(buffer_size=AUTOTUNE)

    return ds

k_train_ds = get_dataset(k_train)
```

Korištenjem funkcije `prefetch()` čuva sljedeću grupu u buffer-u, sa `buffer_size` od `AUTOTUNE` vrijednosti. Podatkovni set spremamo u varijablu `k_train_ds` koja će biti korištena kao trening skupina.

7. 4 Definiranje mrežne arhitekture

Postupak definiranja mrežne arhitekture predzadnji je korak učenja modela. Postoje beskonačno mnogo kombinacija slojeva gdje svaka kombinacija može bolje ili lošije utjecati na proces učenja. Za svrhu ovog rada korišteno je učenje s prijenosom znanja, specifično VGG16 model koji se sastoji od 16 slojeva te je treniran na ogromnim količinama podataka. Arhitektura mreže sastoji se od: 16 slojeva VGG16 modela, 3 sloja pred procesiranja slikovnih podataka kao što su rotacija i kontrast, dva potpuno povezana sloja od koji je jedan izlazni te sloj normalizacije podataka. Slojevi su linearno posloženi koristeći se metodom `tf.keras.Sequential`. Slojevi su podijeljeni na 3 skupa:

- `dataAugmentation` – sloj zadužen za pred procesiranje slikovnih podataka
- `baseModel` – sadrži VGG16 slojeve, težinske vrijednosti bazirane su na `imagenet`
- `Ostali` – ručno definirani slojevi za učenje modela

Nakon što su slojevi linearno posloženi potrebno je definirati funkciju za prilagođavanje težinskih vrijednosti (*eng. Optimizer).* Za svrhe ovoga rada korišten je `RMSProp`. Uz funkciju

prilagođavanja težinskih vrijednosti potrebno je odrediti funkciju gubitka (*eng. Loss Function),* obično uz `RMSProp` koristi se `categorical_crossentropy` te je za svrhu ovoga rada iskorištena ista. Nakon definiranja potpuno povezanih slojeva potrebno je definirati njihove aktivacijske funkcije. Ulazni potpuno povezani sloj koristi `relu` aktivacijsku funkciju dok izlazni potpuno povezani sloj koristi `softmax` aktivacijsku funkciju. Način na koji želimo pratiti ponašanje modela tj. smanjivanje greške kroz uzastopne iteracije definira se pomoću parametra `metrics`. Najčešće se koristi `auc` (*eng. Area under curve)* u pozadini s trapezoidnim pravilom. Metoda vraća model spreman za učenje.

```
def make_model():
    with tf.device('/cpu:0'):
dataAugmentation = tf.keras.Sequential([
tf.keras.layers.experimental.preprocessing.RandomContrast(0.22
, seed=SEED),
tf.keras.layers.experimental.preprocessing.RandomFlip(
"horizontal", seed=SEED),
tf.keras.layers.experimental.preprocessing.RandomFlip(
"vertical", seed=SEED)])
baseModel = tf.keras.applications.VGG16(
input_shape=(*IMG_DIM, 3),
include_top = False,
weights="imagenet")
baseModel.trainable = True
model = tf.keras.Sequential([
dataAugmentation,
baseModel,
tf.keras.layers.GlobalAveragePooling2D(),
tf.keras.layers.Dense(16, activation="relu"),
 tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dense(CLASSES_NUM, activation="softmax")])
model.compile(optimizer=tf.keras.optimizers.RMSprop(),
                loss="categorical_crossentropy",
                metrics=tf.keras.metrics.AUC(name="auc"))
    return model
```

7. 5 Definiranje stope učenja

Stopa učenja predstavlja jedan od najvažnijih hiperparametara vezanih za arhitekturu mreže. Koristi se za optimizaciju modela kako bi se prilagodio brzini učenja. Stopa učenja definira veličinu koraka koji se uzima u smjeru optimizacije. Brzina konvergencije također se mijenja pomoću stope učenja. Stopa učenja je ključni faktor koji definira točnos predikcije modela.

7. 6 Proces učenja modela

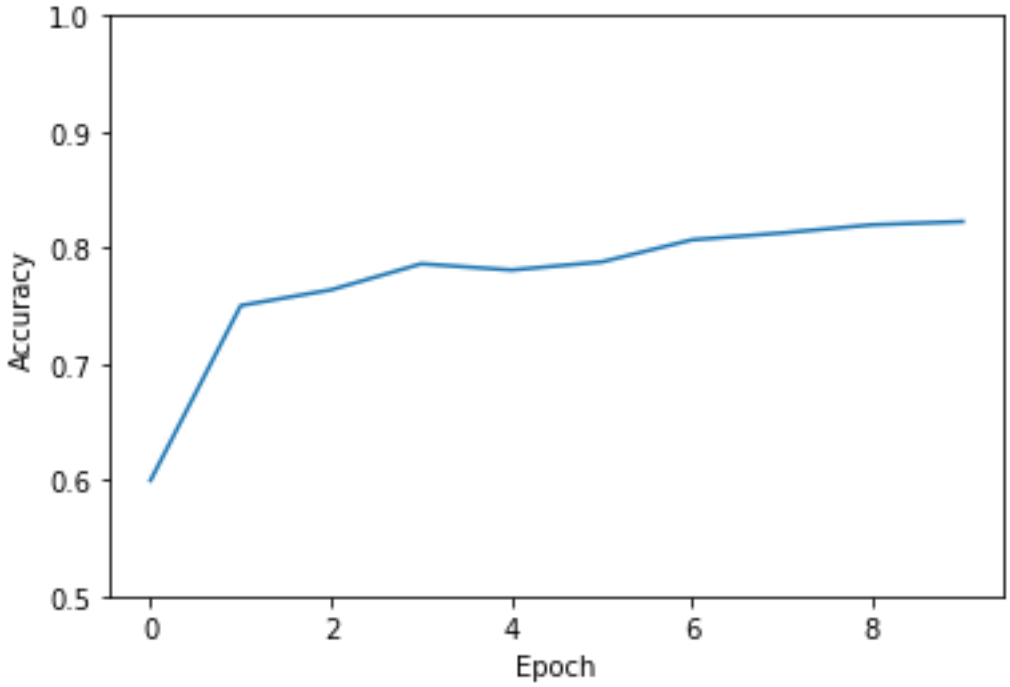
Nakon što su svi podatci obrađeni te arhitektura modela napravljena potrebno je arhitekturu modela dodati kao varijablu u programskom jeziku Python. Nakon što je varijabla `model` potrebno je nad njom pozvati metodu `fit` koja kao parametre uzima obrađeni podatkovni set, broj epoha te niz metoda za povratno manipuliranje težinskim vrijednostima. Varijabla `checkpoint` služi za slušaj da model prestane učiti tj. ako vrijednosti koje prate učinkovitost modela naglo padaju prekidaju učenje te spreme težinske vrijednosti za zadnji najbolji rezultat.

```
with strategy.scope():

    model = make_model()

checkpoint=tf.keras.callbacks.ModelCheckpoint(filepath=
MAIN_DIR, verbose = 1, save_best_only = True)

history=model.fit(k_train_ds,epochs=10,
callbacks=[early_stopping_cb, lr_scheduler, checkpoint])
```

Nakon što je učenje završilo ostaje nam model s spremljenim najboljim težinskim vrijednostima te je moguće izvršenje predikcije u relativno brzom vremenu zbog karakteristike spremanja najboljih težinskih vrijednosti.

7. 7 Rezultati modela

Nakon što je model prošao kroz 10 epoha treninga rezultati modela su spremni za vizualni prikaz.



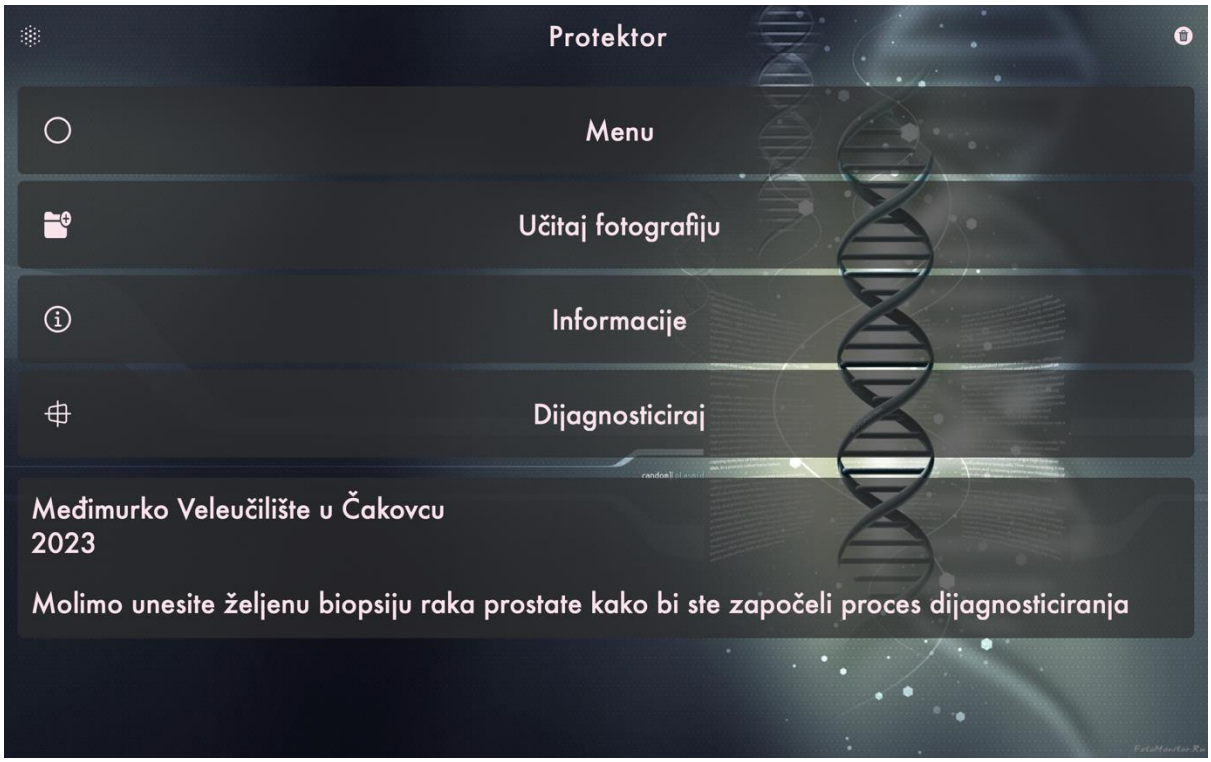
Slika 18. Grafički prikaz efikasnosti modela po epohi

[Izvor: Autor]

8. Grafičko sučelje

Nakon što je model spreman tj. naučen, model se pakira u vanjski paket kako bi se omogućilo korištenje kroz macOS aplikaciju. Grafičko sučelje implementirano je koristeći SwiftUI biblioteku koja omogućuje jednostavno i efikasno generiranje vizualnih elemenata kroz deklarativno programiranje. Sučelje je inspirirano jednostavnošću te intuitivnošću.

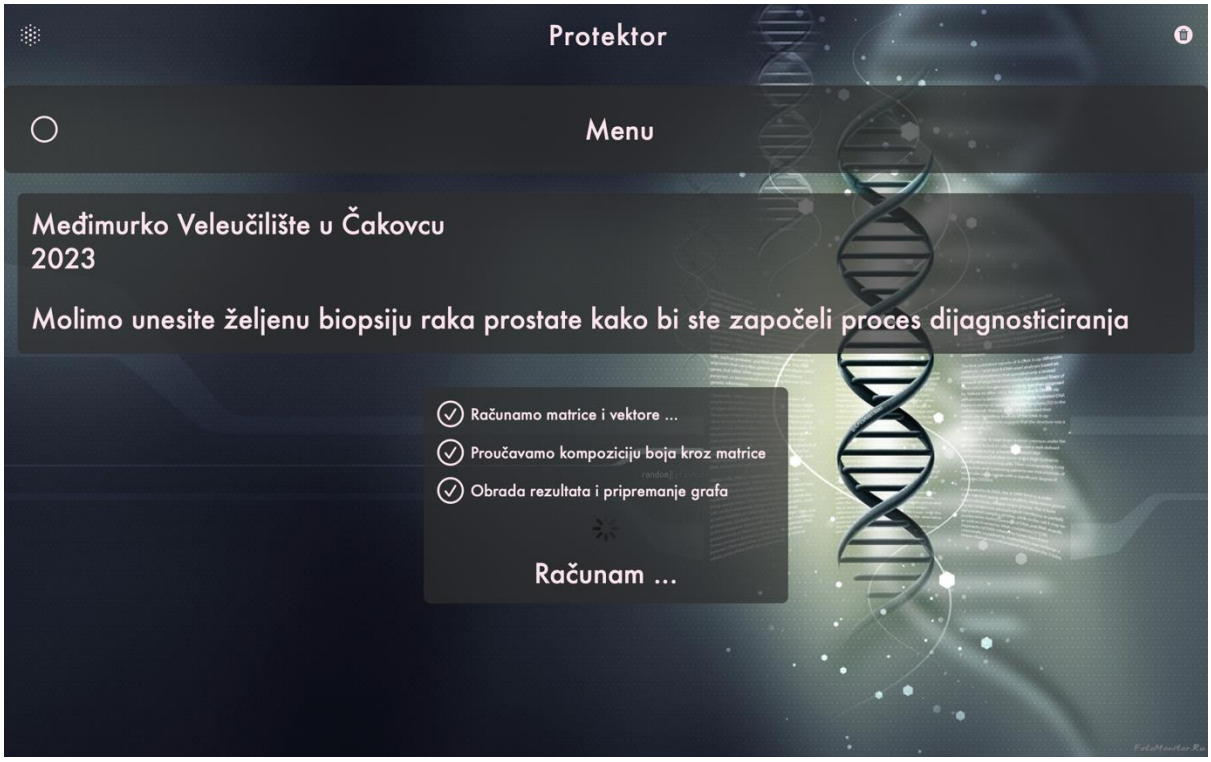
Aplikacija se sastoji od početnog ekrana na kojem je prikazan meni za odabir konačnog broja akcija koje aplikacija nudi. Meni se proširuje na klik te animira ostale elemente menija.



Slika 19. Grafički prikaz otvorenog menija

[Izvor: Autor]

Nakon što se u aplikaciju učita željena fotografija na gumb označen „Dijagnosticiraj“ pokreće se proces predikcije. Proces traje par sekundi zbog kompleksnosti izvođenja operacije.



Slika 20. Prikaz izvođenja predikcije

[Izvor: Autor]

Nakon što je model izvršio predikciju, aplikacija preuzima podatke vraćene kao niz decimalnih brojeva. Podatci se obrađuju te prikazuju kao graf radi jednostavnije prikaza rezultata.

Slika 21. Grafički prikaz predikcije

[Izvor: Autor]

9. ZAKLJUČAK

Ovaj rad je usredotočen na razvoj modela dubokog učenja za predviđanje raka prostate na temelju biopsije prostate. Koristeći podatke iz dostupne baze podataka, uspješno je konstruiran model koji postiže visoku točnost u predviđanju prisutnosti raka prostate. Model koristi duboke neuronske mreže za obradu složenih značajki iz uzoraka biopsije, čime se postiže veća preciznost u predviđanjima.

Rezultati ovog istraživanja pokazuju da je duboko učenje koristan pristup za dijagnostiku raka prostate. Modeli koji se temelje na dubokom učenju mogu biti vrlo precizni i korisni za kliničku praksu, što može dovesti do rane dijagnostike raka prostate i bolje prognoze bolesti. Međutim, kako bismo potvrdili valjanost ovog modela, potrebna su dodatna istraživanja s većim brojem pacijenata i različitim kliničkim varijablama.

U cjelini, ovaj rad predstavlja korak naprijed u razvoju modela dubokog učenja za dijagnostiku raka prostate i potiče daljnja istraživanja u ovoj oblasti.

Ako želimo napredovati kao civilizacija nužno je koristiti sve moguće alate koje nam omogućuju napredak. Stoga je potrebno maknuti stigmu o umjetnoj inteligenciji te iskoristiti sve njene mogućnosti.

9. LITERATURA

[1] Duboko učenje: https://www.ibm.com/topics/deep-learning
[2] Xcode: https://developer.apple.com/documentation/xcode_release_notes/xcode_10_2_1_release_notes
[3] Swift: https://docs.swift.org/swift-book/
[4] Tensorflow: https://www.tensorflow.org/
[5] Anaconda: https://www.anaconda.com/products/distribution
[6] Strojno učenje: https://en.wikipedia.org/wiki/Machine_learning
[7] Duboko učenje: https://books.google.hr/books?hl=en&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=deep+learning&ots=MNU-bptFUT&sig=bxcQnBLS5TsYfeqxkcSX6ljY-QA&redir_esc=y#v=onepage&q=deep%20learning&f=false
[8] Scikit-Learn & Tensorflow: Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems
[9] Keras: https://keras.io/api/
[10] Jupyter Notebook: https://docs.jupyter.org/en/latest/
[11] Rak Prostate: https://www.kaggle.com/code/tanulsingh077/prostate-cancer-in-depth-understanding-eda-model
[12] Učenje s prijenosom znanja: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

POPIS SLIKA: