Integracija tehnologije TensorFlow pri razvoju programskih proizvoda

Gatarić, Jakov

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Jakov Gatarić

Integracija tehnologije TensorFlow pri razvoju programskih proizvoda

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Jakov Gatarić

Matični broj: 46225/11–IZV

Studij: Informacijski sustavi

Integracija tehnologije TensorFlow pri razvoju programskih proizvoda

ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Stapić Zlatko

Varaždin, rujan 2022

Jakov Gatarić

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom završnom radu opisuje se proces kreiranja programskog rješenja za prepoznavanje životinja sa slike pomoću integracije tehnologije TensorFlow. Prije početka izrade programskog rješenja, objašnjen je teorijski dio, odnosno samo učenje nove tehnologije. Za ovaj rad potrebno je znanje strojnog učenja te TensorFlow biblioteke, koja se kroz ovaj rad detaljnije tumači. Teorijski dio rada sastoji se od opisa TensorFlow biblioteke, zašto se on koristi, elaboracije na temu strojno učenje, razrade arhitekture aplikacija koje koriste TensorFlow okvir te sve cjelovito zaokružiti s prikazom primjene navedenih tehnologija. Praktični dio završnog rada se sastoji od izrade programa za detekciju životinja sa slike, tj. tijekom prikaza određene slike, program vraća naziv životinje koja se nalazi na istoj slici. Praktični dio detaljno je dokumentiran kako bi svatko razumio što se u kojem trenu događa. Za izradu programskog rješenja koristi se programski jezik C# i TensorFlow biblioteka.

Ključne riječi:

Strojno učenje, TensorFlow, C#, .NET, prepoznavanje životinja, klasifikacija, razvoj programskih proizvoda

2. Sadržaj

1. Uvod

Umjetna inteligencija (UI) (eng. Artificial Intelligence AI), naziv je kojim označavamo neživi sustav sa sposobnošću snalaženja u nekoj novoj situaciji, odnosno okruženju. Prilikom govora o umjetnoj inteligenciji, obično se misli na računalne sustave ili robote (Raschka S., 2017.) .

Zbog ubrzanog rasta broja izumrlih i ugroženih vrsta životinja, ali i manjka sati provedenih u prirodi, polako zaboravljamo kako određene životinje izgledaju. U ovom radu prikazano je programsko rješenje pomoću kojeg će umjetna inteligencija prepoznati životinju sa slike te istu ispisati. Na ovaj način možemo naučiti djecu kako izgledaju neke životinje. Također se može koristiti za igru i proširenu stvarnost u kojoj se opisuju razni podaci o određenoj životinji. Kroz ovaj rad obuhvaćena je tematika različitih oblika strojnog učenja, ali glavni fokus je na algoritmu sa zadaćom prepoznavanja životinja sa slike. U radu je prikazana općenita skupina životinja (pas, mačka, riba), a ne točna pasmina životinje. Opisane su različite vrste strojnog učenja, različiti algoritmi i dani neki primjeri. Implementacija programskog rješenja detaljno je opisana te je popraćena rezultatima treninga. Cilj završnog rada je opis i izrada modela koji prepoznaje životinju sa slike i ispisuje određene informacije temeljem prepoznavanja životinje sa slike. Učestalim korištenjem Google Lens moguće je primijetiti koliko je ustvari ta aplikacija moćna te sam se sam želio iskušati u izradi jednostavnije aplikacije za prepoznavanje objekata sa slike.

Rad je podijeljen u dva osnovna dijela: teorijski i praktični. U teorijskom dijelu objašnjeni su neki od pristupa strojnog učenja i rasprostranjenost samog strojnog učenja u svakodnevnom životu. Praktični dio ovog rada bazira se na problemu prepoznavanja životinja sa slike i njegovom programskom rješenju u C#-u. Kroz poglavlja završnog rada navode se i detaljnije objašnjavanju metode korištene za dobivanje rješenje našega problema kao i nešto detaljnije o samom strojnom učenju u kojem se kroz potpoglavlja analizira primjena strojnog učenja, razlika strojnog učenja i umjetne inteligencije, vrste strojnog učenja, neuronske mreže te TensorFlow. Završni dio rada obuhvaća praktični primjer programskog rješenja problematike prepoznavanja životinje sa slike u programskom jeziku C# uz detaljna objašnjenja istog. Nakon praktičnog dijela slijede zaključak s kratkim osvrtom na cijeli rad.

2. Metode i tehnike rada

Prilikom rada na ovom završnom radu, korištena je aplikacija Visual Studio za rad u programskom jeziku C#. Microsoft Visual Studio je integrirano razvojno okruženje koje je napravio Microsoft i koristi se za različite vrste razvoja softvera. Sadrži alate za dovršavanje, prevoditelje i druge značajke za olakšavanje procesa razvoja softvera. C# je moderan, objektno orijentiran i siguran programski jezik. C# programerima omogućuje izradu mnogih vrsta sigurnih i ogromnih aplikacija koje se izvode u .NET-u. Cijeli projekt rađen je na operacijskom sustavu Windows 10.

Teorijski dio je izrađen pomoću metoda sinteze kako bi se složena znanstvena istraživanja predstavila i objasnila putem jednostavnih sudova pomoću kojih bi rad mogli pratiti ljudi koji nisu upoznati s temom ili tehnologijom općenito. Prilikom istraživanja literature za izradu teorijskog dijela većinski je korištena literatura stranih autora uz nekoliko nadopuna s hrvatskim autorima. Osim knjiga temeljenih na strojnom učenju i TensorFlowu, konzultirao sam relevantne sekundarne izvore poput znanstvenih i stručnih članaka.

2.1. ML.NET

Upotrebom ML.NET [1] moguće je kreirati prilagođene modele strojnog učenja koristeći C# ili F# bez potrebe za napuštanjem .NET ekosustava. Zbog toga, sve znanje, vještine, kôdovi i biblioteke u .NET okruženju moguće je jednostavno proširiti dodavanjem strojnog učenja u web, mobilne ili stolne aplikacije (Microsoft .NET, 2022).

ML.NET je dizajniran kao proširiva platforma tako da je moguće koristiti druge popularne okvire (TensorFlow, ONNX, Infer.NET i druge) i imati pristup još većem broju scenarija strojnog učenja. Osim različitih okvira, ML.NET posjeduje mogućnost pokretanja putem više platformi poput Windows, Linux i MacOS.

[1] https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet

2.2. ML.NET Model Builder

ML.NET Model Builder [2] pruža lako razumljivo vizualno sučelje za izgradnju, obuku i implementaciju prilagođenih modela strojnog učenja. Model Builder podržava AutoML koji automatski istražuje različite algoritme i postavke strojnog učenja kako bi pomogao pronaći onaj algoritam koji najbolje odgovara određenom scenariju (.NET Microsoft, 2022).

Model Builder je proširenje Visual Studia koje omogućuje programeru bez znanja o strojnom učenju da koristi ovo vizualno sučelje za obuku i korištenje modela, ali i generiranje programskog kôda za obuku i validaciju. Iako Model Builder generira kôd, programer se ne ograničuje prilikom modificiranja istog. Generirani kod se može mijenjati korištenjem drugih algoritama, mijenjanjem parametara, rada s podacima i slično.

[2] https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet/model-builder

3. Strojno učenje

Trenutno živimo u razdoblju u kojem se podaci mogu prikupiti na različite načine te ih ima u izobilju. Pomoću algoritama i strojnog učenja (eng. machine learning) te iste podatke moguće je pretvoriti u znanje. Posljednjih godina razvijaju se mnoge moćne biblioteke otvorenog kôda koje pomažu brže i lakše savladati ovu široku i rastuću granu informatike, stoga je trenutno odlično vrijeme za početak učenja strojnog učenja i učenja kako koristiti algoritme za uočavanje obrazaca u podacima te predviđanje budućih (Shalev-Shwartz S., 2014).

Ovo je glavni dio rada u kojem se detaljnije razrađuje tema, pojašnjavaju istraživanja, prikazuju rezultati i slično. Zbog složenosti cjeline, na samom početku poglavlja dani je kratki opis strukture poglavlja kako bi čitatelj/čitateljica mogla lakše pratiti sve informacije s kojima se možda prije nije susreo/susrela.

3.1. Osnove strojnog učenja

U trenutnom razdoblju, uz pomoć interneta, postoji resurs kojeg imamo u ogromnoj količini, a to je velika količina strukturiranih i nestrukturiranih podataka. Strojno učenje relativno je novo područje informatike koje je evoluiralo iz polja umjetne inteligencije. Sadrži algoritme za samostalno učenje koje je izvlačilo znanje iz podataka kako bi napravili određeno predviđanje. U prošlosti, ljudi su trebali ručno izvoditi pravila, računice i raditi modele pomoću analize podataka, no danas je to puno učinkovitije uz strojno učenje. Model strojnog učenja svakim novim podatkom postepeno poboljšava svoju performansu predviđanja i donošenja odluka s obzirom na dobivene podatke (Smola A., 2008).

U osnovi, strojno učenje zahtjeva da računala otkrivaju kako izvršiti određene zadatke, a da nisu izričito isprogramirani za rješavanje istih. „Drugim riječima, umjetnost strojnog učenja je smanjiti niz prilično različitih problema na skup prilično uskih prototipova“ (Smola A., 2008). Kada je zadatak rješavanje jednostavnih zadataka kojima je moguće programirati algoritme za dobivanje rješenja, strojno učenje nije potrebno. Ono nam je potrebno prilikom rješavanja naprednijih zadataka koji stvaraju probleme čovjeku prilikom pisanja algoritama za njegovo rješavanje. Kod takvih problema moguće su dvije opcije:

1. Programer izvodi svaki mogući scenarij i svaki potreban korak
2. Stroj razvija vlastiti algoritam

Naravno, opcija gdje stroj sam razvija algoritam je mnogo učinkovitija te se ona najčešće i izvodi. Prilikom rješavanja nekog problema, pa čak i nekih jednostavnih, uvijek postoji više načina za dolazak do rješenja. Kada se nađemo u situaciji gdje imamo veliki broj potencijalnih točnih rješenja, označimo točne odgovore kao valjane. Ti se podaci tada koriste za učenje računala za poboljšanje algoritama koji utvrđuju točna rješenja.

3.2. Primjena strojnog učenja

Umjetna inteligencija prisutna je svuda oko nas, iako toga nismo svjesni. Jedna od popularnih aplikacija umjetne inteligencije je strojno učenje u kojoj računalo ili softver radi i uči putem spoznaje. Strojno učenje je prisutno već dugi broj godina, a u posljednjih nekoliko godina se ubrzano razvija zbog svoje široke koristi. Postavlja se pitanje „zašto trebamo strojno učenje ako možemo isprogramirati program koji će riješiti naš problem?“. Odgovor leži u dva aspekta određenog problema koji mogu zahtijevati korištenje programa koji uče i poboljšavaju se na temelju svoj „iskustva“: složenost problema i potreba za prilagođavanjem (Alpaydin E., 2021).

Postoje mnoge aktivnosti koje mi ljudi radimo svakodnevno, ali vrlo teško bismo te iste aktivnosti uspjeli uspješno isprogramirati. Dobar primjer toga je vožnja automobila. Programi koji „uče iz iskustva“ postižu sasvim zadovoljavajuće rezultate nakon izlaganja dovoljnom broju primjeraka obuke (Smola, A., 2008).

Drugi razlog korištenja strojnog učenja je analiziranje velikog i kompleksnog broja podataka. U nastavku ćemo spomenuti neke od najpopularnijih primjena strojnog učenja u stvarnom svijetu.

3.2.1. Prepoznavanje slika

Prepoznavanje slika jedna je od najčešćih primjena strojnog učenja. Korist je pronašla u identifikaciji objekata, osoba, mjesta itd. Popularan primjer toga je prijedlog za automatsko označavanje prijatelja na Facebook-u. Kad god učitamo sliku sa svojim prijateljima na Facebook-u, automatski dobivamo prijedlog označavanja s imenom toga prijatelja. Naravno iza toga stoji strojno učenje za otkrivanje i prepoznavanje lica (Potentia, 2021).

Zadatak algoritma strojnog učenja je identificiranje objekta unutar slike i prepoznavanje kojoj kategoriji slika pripada. Kada ljudi vide neki objekt ili scenu, automatski identificiraju objekte kao različite instance i povezuju ih s pojedinačnim definicijama.

Međutim, vizualno prepoznavanje je vrlo složen zadatak za strojeve koji zahtijeva značajnu procesorsku snagu. Dok su se različite metode oponašanja ljudskog vida razvijale

tijekom vremena, zajednički cilj prepoznavanja slike je klasifikacija detektiranih objekata u različite kategorije (Veel M., 2022).

Zbog velike potražnje ova vrsta strojnog učenja je najrazvijenija, a u nastavku je moguće vidjeti kako funkcionira prepoznavanje slika.

Prvi korak je prikupljanje i organiziranje podataka. Organizacija podataka znači klasificiranje svake slike i razlikovanje njezinih fizičkih karakteristika. Računala percipiraju sliku kao vektorsku ili rastersku sliku, dakle, nakon što su stvoreni konstrukti koji prikazuju objekte i značajke slike, računalo ih analizira (Yegulalp S., 2022).

Iz ovoga možemo zaključiti kako je prikupljanje i ispravno organiziranje podataka ključno za treniranje modela prepoznavanja slike. Ako kvaliteta podataka nije na zadovoljavajućoj razini, algoritam neće moći prepoznati ispravne uzorke u kasnijoj fazi.

Drugi korak procesa prepoznavanja slike je izgradnja referentnog modela. Algoritam za klasifikaciju se mora pažljivo trenirati, inače neće moći ispuniti svoju funkciju. Algoritam koristi skupove podataka dubokog učenja (eng. deep learning) za razlikovanje uzoraka na slikama. Ovi se skupovi podataka sastoje od stotina tisuća označenih slika. Algoritam pregledava te skupove podataka i uči kako izgleda slika određenog objekta (Yegulalp S., 2022).

Posljednji korak izgradnje algoritma za prepoznavanje slika je njegovo testiranje. Prilikom testiranja važno je pronaći slike koje nisu bile dio treninga. Odlična praksa je iskoristiti 80%-90% slika tijekom učenja, a ostatak iskoristiti za testiranja. Učinkovitost modela mjeri se na temelju skupa parametara koji pokazuju postotak točnosti po testnoj slici, zajedno s netočnim identifikacijama.

3.2.2. Prepoznavanje govora

Prepoznavanje govora proces je pretvaranja glasovnih uputa u tekst, a poznato je kao „Speech to text“. Trenutno najpoznatiji primjeri su Google-ovo glasovno pretraživanje i korištenje virtualnog asistenta koji koristi tehnologiju prepoznavanja govora kako bi slijedili glasovne upute (Potentia, 2021).

Pomoću prepoznavanja govora možemo govoriti te se ono pretvara u poruku koja je spremna za slanje, ali i za pisanje natuknica prilikom učenja ili tijekom predavanja.

Prepoznavanje govora djeluje pomoću sustava programa i algoritama strojnog učenja koji su u međusobnoj interakciji, poput modela izgovora i akustičkih modela koji „čuju“ i prepoznaju govorni jezik, kao i jezičnih modela koji određuju što je rečeno u pokušaju da se formira najvjerojatnije značenje (Defined.AI, 2022).

Algoritam prepoznavanja govora radi tako da snima ljudski govor, sprema kao skup podataka kojeg rastavlja na manje dijelove informacija te uči prepoznavati i interpretirati govorne obrasce.

Stvari se kompliciraju kada program ne radi u „laboratorijskim uvjetima“, odnosno kada nije tišina i naš govor nije razumljiv. Kada se nalazimo u bučnom okruženju, npr. u tramvaju, vlaku, na nekom okupljanju, strojevi trebaju izolirati glas od neželjene buke zbog točnijeg prepoznavanja riječi. Također, strojno učenje mora uzeti u obzir naglaske, lokalne izraze i različite načine govorenja istih stvari unutar svakog pojedinog jezika. Dodavanjem subjektivnih elemenata poput namjera ili osjećaja, sustav za strojno učenje postaje još kompliciraniji u svrhu da radi najbolje što može (Defined.AI, 2022).

Samim korištenjem ove značajke ona postaje bolja i točnija zbog konstantnog učenja našeg vokabulara, naglaska i načina pričanja. U nastavku su nabrojeni i opisani neki od ključnih primjera tehnologije prepoznavanja govora iz stvarnog života.

Pozivni centri

U samim počecima, implementacija umjetne inteligencije bila je usmjerena na njegovu primjenu u kontekstu uštede troškova, no danas se koristi za poboljšanje korisničkog iskustva i vrijednosti kupaca. Od usmjeravanja pozivnog centra do odgovaranja na osnovne i najčešće upite, tehnologija prepoznavanja govora postala je neizbježan dio poslovanja za većinu organizacija (Defined.AI, 2022).

Bankarstvo

Glavni prioriteti klijenata unutar bankarstva su sigurnost i korisničko iskustvo. Korištenje umjetne inteligencije u bankarstvu može pomoći i klijentima i bankama. Sa sigurnosne strane, banke koriste prepoznavanje govora kako bi omogućile plaćanje unutar mobilnog i online bankarstva. Kako bi banke ojačale višefaktorski sustav, često se prepoznavanje glasa kombinira s biometrijom lica. Logika je da dok kada korisnik govori provjerava se autentičnost govora, a algoritam za prepoznavanje lica prati pomicanje usana (Defined.AI, 2022).

S korisničke strane, kao i u pozivnim centrima, prepoznavanje govora koristi se za rješavanje jednostavnih ili najčešćih upita bez potrebe za čekanje u redovima za razgovor s ljudskim agentom.

3.2.3. Predviđanje prometa

Kada putujemo na neko nepoznato mjesto, velika je vjerojatnost da ćemo koristiti Google Maps prilikom putovanja. Ono nam pokazuje put s najidealnijom rutom i predviđa moguće

prometne uvjete poput zastoja, radova na cesti i slično. Google Maps radi na sljedeći način. Prati lokaciju vozila u stvarnom vremenu pomoću svoje aplikacije i senzora (Potentia, 2021).

Svi korisnici pomažu u poboljšanju ove aplikacije. Aplikacija uzima korisnikove informacije o putovanju i šalje ih u svoju bazu podataka gdje se informacije koriste za učenje i predviđanje budućih prometnih uvjeta.

Predviđanje prometa znači predviđanje količine i gustoće prometnog toka u svrhu upravljanja kretanjem vozila, smanjenja gužvi i generiranja optimalne rute s najmanjim utroškom vremena ili energije/goriva (Altexsoft, 2022).

Danas postoje različite tehnike strojnog učenja koje mogu obraditi ogromne količine povijesnih podataka i podataka u stvarnom vremenu koje se koriste za predviđanje toka, gustoće i brzine prometa. Kako bi razumjeli rad algoritama potrebno je pogledati koji su podaci potrebni za predviđanje prometa i odakle se oni dobivaju.

Prije svega, potrebno je imati detaljnu kartu s cestovnom mrežom i povezanim atributima. Povezivanje s globalnim pružateljima kartografskih podataka poput Google Maps, TomTom, HERE ili OSM izvrstan je način za dobivanje potpunih i ažurnih informacija (Altexsoft, 2022).

Nakon detaljnih karata, potrebno je prikupiti povijesne i trenutne informacije vezane uz promet kao što su broj vozila koji prolaze određenom točkom, njihova brzina i vrsta (teretna vozila, laka vozila itd.). Ove podatke moguće je prikupiti pomoću sljedećih uređaja: detektori petlja, kamere, senzori pokreta, radari i pametni telefoni (Altexsoft, 2022).

Sve ove informacije najlakše je dobiti od navedenih pružatelja usluga. Drugi važni podaci vezani za promet su informacije o incidentima, odnosno zatvaranju cesta i radovi na cesti. Potrebno je spomenuti da su od velike koristi i podaci o vremenu. Trenutni i prognozirani podaci o vremenu su također ključan dio u predviđanju vremena putovanja jer meteorološki uvjeti utječu na situaciju na cesti i brzinu vožnje. Posljednja vrsta podataka su dodatni podaci o stanju na cestama. Ovdje se misli na sve vanjske izvore podataka koji pružaju važne informacije koje utječu na promet. Kao primjer toga su objave na društvenim mrežama o sportskom događaju u tom području (zatvorena cesta zbog maratona/vožnje biciklista), građanskim prosvjedima, pa čak i policijskim skenerima i nesrećama.

3.2.4. Preporuka proizvoda

Prema Duggal N., preporuka proizvoda jedna je od najpopularnijih i najpoznatijih primjena strojnog učenja. Ona je jedna od glavnih značajki u raznim e-trgovinama danas. Ovaj primjer možemo uočiti tako da upišemo određeni proizvod u tražilicu na Amazonu, stavimo ga

u košaricu ili čak i kupimo. S vremenom bismo počeli dobivati oglase za isti ili slični proizvod na ostalim stranicama tijekom surfanja internetom, a svemu tome je krivo strojno učenje koje prati naše ponašanje.

Mehanizam za preporuku proizvoda je sustav za filtriranje informacija koji učitava informacije prilagođene interesima korisnika ili povijesti ponašanja na stavci. Ovaj mehanizam je vrhunski marketinški alat za e-trgovine koji donosi veći profit, prodaju i prihode. To je glavni razlog rasprostranjenosti ovog alata te će njegova korist samo rasti. Kako bi alat bio koristan, on mora biti fleksibilan ponašanju novih korisnika. Također bi trebao moći djelovati u dinamičnom okruženju, pružajući korisnicima pravodobne informacije o posebnim ponudama, promjenama asortimana i cijenama (Duggal N., 2022).

Kako bi kupcima pružili preporuke za usluge ili proizvode, koriste se algoritmi strojnog učenja koji čine proces predviđanja stavki točnijim. Algoritmi se mijenjaju temeljem podataka dobivenih od sustava preporuka. Algoritmi strojnog učenja za sustave preporuke dijele se u dvije kategorije: kolaborativno filtriranje i filtriranje temeljeno na sadržaju. Moderni sustavi preporuka kombiniraju oboje. Filtriranje temeljeno na sadržaju uzima u obzir sličnost atributa proizvoda, a kolaborativno filtriranje računa sličnost iz interakcija kupaca (Schiavini, 2019).

Primjer filtriranja temeljenog na sadržaju bio bi sljedeći: kupujemo plastičnu masku za mobitel te pomoću sustava preporuke preporučena nam je silikonska maska. Što se tiče kolaborativnog filtriranja, ako ponovno kupujemo masku za mobitel preporuka nam postaje zaštitno kaljeno staklo. Na velikom broju stranica kolaborativno filtriranje je vidljivo u obliku: „Kupci koji su kupili proizvod X također su kupili i sljedeće“.

Uz toliko informacija na Internetu i tolikom broju ljudi koji ih koriste, postalo je vrlo važno za organizacije da ulažu vrijeme i novac u alate umjetne inteligencije koji odgovaraju tržišnim potrebama i ukusima.

3.2.5. Samovozeći automobili

Definitivno najzanimljivija i najuzbudljivija primjena strojnog učenja su autonomna vozila. Strojno učenje ima glavnu ulogu kod takvih automobila, a najbolji primjer takvih automobila je tvrtka Tesla. Koristi se metoda učenja bez nadzora prilikom otkrivanja ljudi, prometnih znakova i ostalih predmeta tijekom vožnje. Na internetu je moguće vidjeti videozapise kako program prepoznaje i zapisuje podatke u stvarnom vremenu te ovisno o podatku automobil skreće, koči i slično.

Samovozeći automobil (autonomni automobil ili automobil bez vozača) vozilo je koje koristi kombinaciju senzora, kamera, radara i umjetne inteligencije za putovanje između

odredišta bez ljudskog operatera. Umjetna inteligencija pokreće okvire samovozećih automobila. Inženjeri samovozećih automobila koriste veliki broj informacija iz sustava za prepoznavanje slika, zajedno s umjetnom inteligencijom i neuronskim mrežama, kako bi sastavili okvire koji mogu voziti samostalno. Neuronske mreže razlikuju uzorke u podacima koji se unose u izračune umjetne inteligencije. Ti podaci se dobivaju pomoću kamera u vozilu. Neuronske mreže moraju prepoznati automobile, prometne signale i znakove, drveće, ljude koji hodaju i različite dijelove bilo kojeg dinamičnog okruženja vožnje (Lutkevich B., 2019).

Googleov projekt Waymo primjer je samovozećeg automobila koji je gotovo u potpunosti autonoman. Naime, zahtijeva prisutnost ljudskog vozača, ali samo da nadjača sustav kada je to potrebno. Drugim riječima, nije samovozeći u najčišćem smislu, ali se može sam voziti u idealnim uvjetima te ima visoku razinu autonomije. Mnogi automobili koji su dostupni potrošačima imaju nižu razinu autonomije, ali još uvijek imaju neke značajke samostalne vožnje. Značajke samostalne vožnje koje su dostupne u mnogim serijskim automobilima od 2019. godine kako napominje Lutkevich su sljedeće:

- Upravljanje bez ruku centrira automobil bez ruku vozača na upravljaču, ali vozač je i dalje dužan obraćati pozornost
- Prilagodljivi tempomat koji automatski održava udaljenost između vozačevog automobila i automobila ispred kojeg se nalazi
- Upravljanje centriranjem na kolničkoj traci intervenira kada vozač prijeđe oznake trake automatski gurajući automobil prema suprotnim oznakama trake

Googleov Waymo sklopio je partnerstvo s Lyftom kako bi ponudio potpuno autonomnu komercijalnu uslugu dijeljenja prijevoza pod nazivom Waymo One. Korisnici mogu pozvati samovozeći automobil da ih doveze do odredišta i vožnjom daju povratnu informaciju Waymu. Automobili još uvijek sadrže sigurnosnog vozača u slučaju da treba nadjačati sustav (Lutkevich B., 2019).

3.2.6. Otkrivanje prijevara

Prilikom svakog izvršenja internetske transakcije, postoje različiti načini na koje se može dogoditi lažna transakcija, poput krivotvorenih računa, lažnih podataka i krađe novca usred transakcije.

U borbi s ovim problemom pomaže nam napredna neuronska mreža prilikom provjere radi li se o izvornoj ili lažnoj transakciji. Nakon svake ispravne transakcije, izlaz se pretvara u neku hash vrijednost, a te vrijednosti postaju ulaz kod sljedećeg unosa (Potentia, 2021).

U posljednjih nekoliko godina masovno se razvio sustav e-trgovanja, a samim time i online plaćanje. Otkrivanje prijevara u dinamičnom globalnom poslovnom okruženju s ogromnom količinom prometa i podataka za praćenje može biti vrlo težak posao.

Umjetna inteligencija ima izvrsnu aplikaciju za otkrivanje prijevara zbog svoje zbirke informacija u bankarstvu i osiguranju. Naravno, sve banke i druge financijske institucije suočavaju se s novim izazovima kada je riječ o stvaranju sigurnosti za sebe i svoje klijente. Smanjenje prijevara glavni je cilj banaka zato što one mogu oštetiti njihov ugled, uzrokovati loša korisnička iskustva i napraviti štetu kod zadržavanja kupaca. Tome u pomoć dolazi umjetna inteligencija koja pomaže tvrtkama u poboljšanju unutarnje sigurnosti i pojednostavljenju korporativnih operacija. U otkrivanju prijevara, strojno učenje je skup algoritama umjetne inteligencije obučenih s povijesnim podacima za predlaganje pravila rizika. Moguće je implementirati pravila za blokiranje ili dopuštanje određenih radnji korisnika, poput sumnjive prijave, krađe identiteta ili lažne transakcije. Prilikom obučavanja mehanizma za strojno učenje potrebno je označiti prethodne slučajeve prijevare i uspješne transakcije kako bi izbjegli lažno pozitivne rezultate i poboljšali preciznost pravila o riziku. Što dulje algoritmi rade, prijedlozi pravila će biti točniji (Florian, 2022).

Kada je riječ o obradi velikog skupa podataka, strojevi su daleko brži i efikasniji u obradi istih od ljudi. Kako navodi Florian, ovo su neke od glavnih prednosti u korištenju strojnog učenja za otkrivanje prijevare:

- Brže i učinkovitije otkrivanje. Sustav može brzo prepoznati sumnjive obrasce i ponašanja za koje bi ljudskim agentima možda trebali mjeseci.
- Smanjeno vrijeme ručnog pregleda: Slično prethodnom, količina vremena potrošena na ručno pregledavanje informacija može se drastično smanjiti kada dopustimo strojevima da analiziraju sve podatkovne točke umjesto nas.
- Bolja predviđanja s velikim skupovima podataka: Što više podataka unesemo u stroj za strojno učenje, on postaje uvježbaniji. Drugim riječima, dok veliki skupovi podataka ponekad ljudima mogu predstavljati izazov za pronalaženje obrazaca, zapravo je suprotno sa sustavom koji pokreće umjetnu inteligenciju.
- Isplativo rješenje. Za razliku od zapošljavanja više ljudskih agenata, potreban je jedan sustav strojnog učenja za prolazak kroz sve podatke, bez obzira na količinu.

Možda i najveća prednost strojnog učenja je ta da algoritmima nije potrebna pauza, godišnji odmor ili slično. Napadi i prijevare se mogu dogoditi 24 sata dnevno, 7 dana u tjednu, tako da je sustav uvijek spreman za obranu i održavanje ugleda tvrtke.

3.2.7. Trgovanje na burzi

Iako se možda ne priča toliko o tome, strojno se učenje uveliko koristi kod trgovanja na burzi. Na burzi uvijek postoji rizik od povećanja ili pada vrijednosti dionica, pa se u ovom slučaju koristi dugoročna memorijska neuronska mreža za predviđanje budućih trendova. Ova vrsta umjetne inteligencije se razvija velikom brzinom, postaje sve dostupnija velikim skupinama korisnika i istovremeno revolucionira s tehnološkim napretkom, a sve je to zbog obećavajuće dobre zarade.

Rješenja umjetne inteligencije sposobna su vrlo brzo brojati brojeve i donositi optimalne odluke na temelju velikog broja podataka, što je vrlo primjenjivo na realnost burze. Strojno učenje omogućuje financijskim tvrtkama da dobiju potpunu sliku situacije na burzi uz pomoć dubinske, kontinuirane analize fluktuacije cijena dionica i nestrukturirane obrade podataka. Također se pokazao korisnim u identifikaciji složenih obrazaca trgovanja, donoseći ispravne odluke o prodaji/kupnji u stvarnom vremenu (Datrics, 2022).

Očigledna prednost umjetne inteligencije nad ljudima je nedostatak emocija. Ljudi su notorno loši u trgovanju dionicama iz jednostavnog razloga jer ih svladaju emocije. Dionice padaju pa se masovno rasprodaju zbog panike. Jednako tome, određena dionica dosegne najvišu cijenu ikada koja samo potiče pohlepne ulagače da kupuju na samom vrhu vrijednosti, što najčešće rezultira gubitku novaca. Računala su nemilosrdna i ne postanu tužni kada se tržišta ne kreću u smjeru koji su predvidjeli. Umjesto toga, umjetna inteligencija iz iskustva uči pravila i praksu trgovanja pa koristi to znanje kako bi bolje procijenili buduće promjene u vrijednosti dionica i samim time poboljšali buduće performanse.

Strategije trgovanja pomoću umjetne inteligencije postaju sve sofisticiranija zato što sustavi uče iz vlastitog iskustva. Iz tog razloga danas korisnicima nude značajne prednosti omogućujući:

- Detekciju uzoraka. Umjetna inteligencija analizira povijesne podatke i vidi ponavljajuće obrasce u dinamici cijena dionica kako bi identificirao pravu strategiju za investitora.
- Trgovanje temeljeno na predviđenom raspoloženju. Programi umjetne inteligencije mogu uključiti podatke iz vijesti i društvenih medija u svoje analize, donoseći odluke na temelju mnogo opsežnijeg skupa podataka nego što bi to dopuštala konvencionalna tehnička analiza.
- Brzo trgovanje. Uz brojanje svake milisekunde u trgovanju dionicama, aplikacije za trgovanje s umjetnom inteligencijom mogu uštedjeti vrijeme i novac olakšavanjem trenutnih odluka i radnji.

Sve ove prednosti govore u prilog korištenju umjetne inteligencije prilikom bilo kakvih investicijskih aktivnosti, međutim umjetna inteligencija nije na razini da u potpunosti zamijeni ljude na burzi, barem ne u današnje vrijeme.

3.2.8. Medicina

Mnogi zdravstveni sektori koriste algoritme strojnog učenja za bolje upravljanje. Koriste se za predviđanje vremena čekanja u čekaonicama, naručivanje određenih lijekova te također dolaze u obzir prilikom dijagnosticiranja bolesti (Schmitt M., 2022).

Strojno učenje pomaže u pronalaženju tumora na mozgu i drugim bolestima povezanim s mozgom. Napredak u strojnom učenju nastavit će mijenjati zdravstvenu industriju na bolje i brže otkrivanje bolesti, ali i pomoći u donošenju određenih dijagnoza. U nastavku su ukratko objašnjene tri glavne aplikacije umjetne inteligencije u medicini.

Dijagnosticiranje bolesti

Za ispravno dijagnosticiranje bolesti potrebne su godine medicinskog obrazovanja, ali i tada je dijagnostika naporan i dugotrajan proces. Nažalost, u mnogim područjima potražnja za stručnjacima daleko premašuje raspoloživu ponudu, no strojno učenje je nedavno napravilo veliki napredak u automatskom dijagnosticiranju bolesti, čineći ju jeftinijom i pristupačnijom.

Algoritmi strojnog učenja uče vidjeti uzorke na isti način na koji ih vide liječnici. Jedina razlika je što algoritmima treba mnogo tisuća konkretnih primjera kako bi što točnije naučili. Naravno, primjeri moraju biti digitalizirani, pa je tako strojno učenje osobito korisno u područjima gdje su dijagnostički podaci koje liječnik ispituje već digitalizirani. Neka od tih područja su: otkrivanje raka pluća ili moždanog udara na temelju CT-a, procjena rizika od iznenadne srčane smrti ili drugih srčanih bolesti na temelju elektrokardiograma i MRI slika srca, klasificiranje kožnih lezija na slikama kože i pronalaženje pokazatelja dijabetičke retinopatije u slikama oka (Schmitt M., 2022).

Ubrzani razvoj lijekova

Razvoj lijekova je skup proces. Mnogi analitički procesi uključeni u razvoj lijekova mogu se učiniti učinkovitijima uz strojno učenje koje bi moglo smanjiti godine rada i milijune ulaganja. Strojno učenje se uspješno koristi u sve 4 glavne faze razvoja lijeka: identificiranje ciljeva, otkrivanje kandidata za lijekove, ubrzavanje kliničkih ispitivanja, pronalaženje biomarkera za dijagnosticiranje bolesti (Schmitt M., 2022).

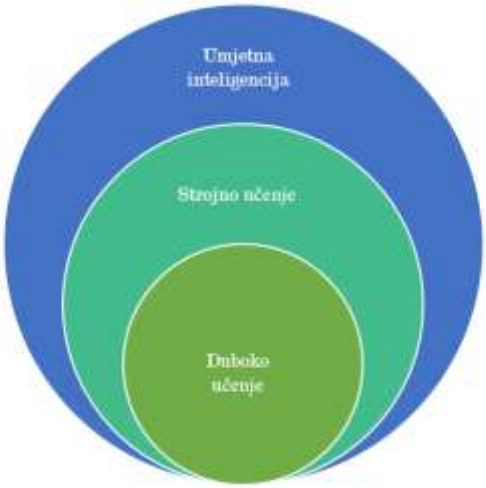
Personalizirani tretman

Različiti pacijenti različito reagiraju na lijekove i rasporede liječenja. Vrlo je teško utvrditi koji čimbenici trebaju utjecati na izbor liječenja, ali uz pomoć strojnog učenja ono može biti automatizirano. Drugim riječima, algoritam strojnog učenja može predvidjeti vjerojatni odgovor pacijenta na određeni tretman. Sustav to uči unakrsnim referencama sličnih pacijenata i usporedbom njihovih tretmana i ishoda (Schmitt M., 2022).

3.3. Razlika umjetne inteligencije i strojnog učenja

Umjetna inteligencija i strojno učenje dio su računalne znanosti koji su međusobno povezani. Ove dvije tehnologije su najmodernije tehnologije koje se koriste za stvaranje inteligentnih sustava, a često se koriste kao sinonim jedne za drugu, što je naravno netočno. Tehnologija iz dana u dan postaje sve više prisutna u našem svakodnevnom životu. Tu činjenicu možemo vidjeti na primjerima društvenih medija (prepoznavanje osoba na slikama na kojima nisu označeni) ili preko razgovora s virtualnim asistentom na vlastitim uređajima (Google asistent, Siri, Alexa itd.). Takva se vrsta tehnologije povezuje s umjetnom inteligencijom, strojnim učenjem, dubokim učenjem i neuronskim mrežama. Iako svi navedeni pojmovi imaju određenu ulogu u ostvarenju tih ciljeva, oni se vrlo često pogrešno koriste u razgovorima, što dovodi do zabuna i nejasnoća.

Najjednostavniji način objašnjenja ovih pojmova bio bi uz primjer ruskih lutaka. Razlog tome je što je svaki od ovih pojmova sastavni dio prethodnog. Drugim riječima, strojno učenje je potpolje umjetne inteligencije. Duboko učenje je potpolje strojnog učenja, a neuronske mreže su potpolje algoritama dubokog učenja. Da bi se duboko učenje smatralo dubokim učenjem, ono mora imati više od tri slojeva neuronskih mreža.



Slika 1: Odnosi umjetne inteligencije, strojnog učenja i dubokog učenja (Berchane N.; 2018; izvor: https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/)

Umjetna inteligencija je grana računalne znanosti koju čini računalni sustav koji pokušava oponašati ljudsku inteligenciju (Shalev-Shwartz S., 2014).

Koristi se za predviđanje, automatiziranje i optimiziranje zadataka koje su u prošlosti radili ljudi, poput prepoznavanja objekata i donošenja određenih odluka. Ono nije potrebno prethodno programirati, već koristi algoritme koji rade pomoću vlastite inteligencije.

Klasično strojno učenje pretežno ovisi o ljudskoj pomoći prilikom učenja. Programeri određuju značajke prema kojima će se početni podaci raspoređivati, što zahtijeva više strukturiranih podataka za učenje. Strojno učenje koristi izrazito velike količine strukturiranih podataka kako bi model strojnog učenja mogao doći do točnih rezultata ili dati predviđanje na temelju istih (Raschka S., 2017.).

Važno je napomenuti da ono radi samo za određene domene, npr. ako kreiramo model strojnog učenja za detekciju životinja, a kod unosa podataka unesemo sliku auta, model će prestati raditi.

Duboko strojno učenje, poznato je još i kao strojno učenje s nadzorom, objasnit ćemo kasnije u radu. U suštini, ono radi na način da može upotrijebiti označene skupove podataka za učenje vlastitog algoritma. Podaci mogu biti i nestrukturirani u sirovom obliku, pa algoritam automatski određuje skup značajki koje se međusobno razlikuju (PCChip, 2021).

3.4. Vrste strojnog učenja

Klasično strojno učenje često se kategorizira prema sposobnosti algoritma da nauči kako postati točniji u svojim predviđanjima. Zanimljivo je naučiti, razumjeti i identificirati vrste strojnog učenja s kojima se možemo susresti.

Za prosječnog korisnika računala to može biti u obliku razumijevanja vrsta strojnog učenja i načina na koji se strojna učenja mogu pokazati u aplikacijama koje sam korisnik koristi. Za programere koji stvaraju ove aplikacije, bitno je poznavati vrste strojnog učenja kako bi za bilo koji zadatak s kojim se mogu susresti mogli izraditi odgovarajuće okruženje za učenje i razumjeti zašto je baš to okruženje koje su izradili uspjelo (Heidenreich H., 2018).

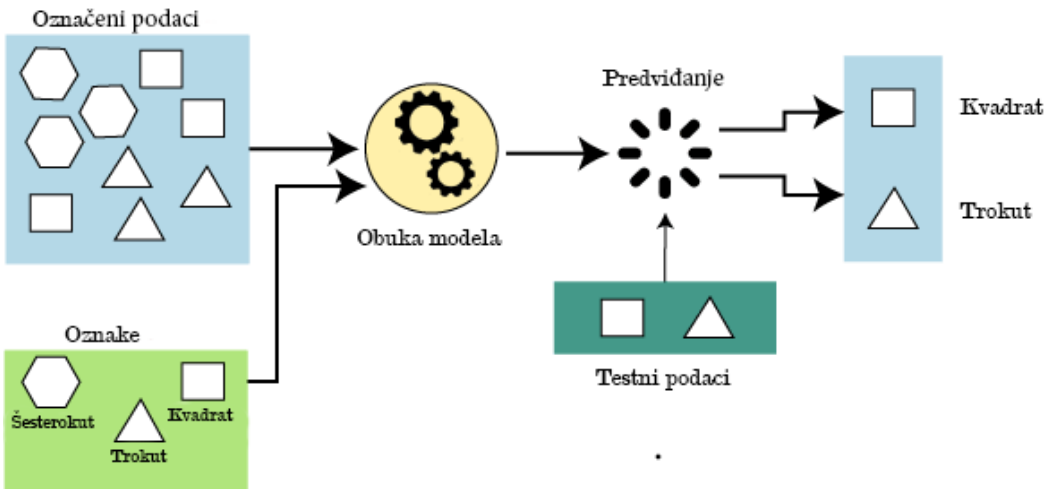
Web stranica JavaTPoint nalaže kako postoje 4 vrste strojnog učenja, a to su sljedeće:

1. Učenje s nadzorom (eng. supervised learning)
2. Učenje bez nadzora (eng. unsupervised learning)
3. Polu-nadzirano učenje (eng. semi-supervised learning)
4. Učenje s pojačanjem (eng. reinforcement learning)

Svaka vrsta strojnog učenje bit će detaljnije objašnjena, isto kao i glavne razlike između njih. Primjerima ćemo potkrijepiti teorijsko znanje i prikazati praktična područja u kojima se određene vrste strojnog učenja primjenjuju.

3.4.1. Učenje s nadzorom

Glavna svrha učenja s nadzorom je naučiti model iz označenih podataka koji nam omogućuje predviđanje neviđenih ili budućih podataka. Ono je najosnovnija vrsta strojnog učenja. Razlog zašto se naziva učenje s nadzorom je taj što se koriste već poznati uzorci sa željenim izlaznim oznakama. Iako podaci moraju biti točno označeni kako bi ova metoda ispravno funkcionirala, ona je vrlo moćna kada se koristi u pravim okolnostima (javaTpoint, 2021).



Slika 2: Strojno učenje s nadzorom (JavaTPoint; 2021; izvor: https://www.javatpoint.com/supervised-machine-learning)

U učenju s nadzorom algoritam prima manji skup podataka za treniranje. Manji skup je obično podskup većeg skupa, ali njegova je zadaća pružiti osnovnu ideju o problemu, rješenju i podacima koji se koriste prilikom učenja. Manji skup je dakle vrlo sličan velikom skupu te služi za generiranje označenih parametara potrebnih za rješavanje problema. Algoritam traži odnose između parametara, odnosno uspostavlja uzročno-posljedičnu vezu između varijabli u skupu podataka (javaTpoint, 2021).

Na kraju obučavanja, algoritam ima ideju o odnosu i načinu rada podataka. Nakon obuke algoritam se primjenjuje na ukupan, tj. cijeli skup podataka te uči na isti način kao što je učio i s manjim skupom. Ovakav pristup znači da će algoritam nastaviti učiti i nakon implementacije, otkrivajući nove odnose između novih podataka (Heidenreich H., 2018).

Kao primjer učenja s nadzorom, uzeti ćemo filtriranje neželjene e-pošte. Model je moguće trenirati pomoću algoritama postavljenih na zbirku podataka označenih e-poruka. Poruke koje su ispravno označene kao neželjene pomažu algoritmu predvidjeti hoće li nova e-poruka pripasti u poželjnu ili nepoželjnu kategoriju. Učenje s nadzorom diskretnim oznakama kategorija, poput našeg primjera, naziva se klasifikacija. Druga vrsta učenja s nadzorom jest regresija koja se koristi za predviđanje kontinuiranih vrijednosti.

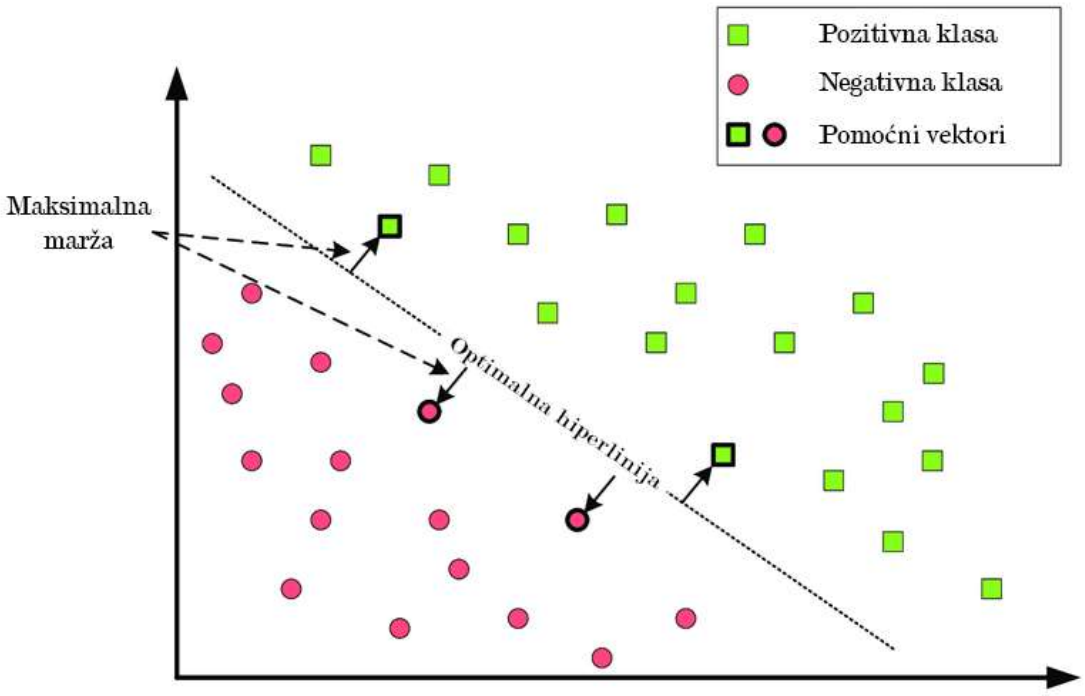
3.4.1.1. Klasifikacija

Klasifikacija je potkategorija učenja s nadzorom u kojem je cilj predvidjeti kategorizaciju novih podataka, na temelju prethodnih opažanja. Te kategorizacije su diskretne, neuređene vrijednosti koje se mogu shvatiti kao određena grupa u instancama (Heidenreich H., 2018).

Primjer nepoželjne e-pošte vrlo je dobar primjer binarne klasifikacije gdje algoritam strojnog učenja detektira uzorke kako bi razlikovao neželjenu od željene e-pošte. Klasifikacija ne mora samo biti binarna, već algoritam može sam dodijeliti oznaku predstavljenom skupu podataka prilikom obuke instanci bez oznaka. Jedan od brojnih primjera klasifikacije u više klasa je prepoznavanje životinja i upravo ćemo njega koristiti u praktičnom dijelu rada. Klasifikacija životinja može biti podijeljena na nekoliko načina:

- Kralježnjaci i beskralježnjaci
- Kopnene i vodene

Kako bi algoritam ispravno i uspješno funkcionirao, potrebno je prikupiti fotografije za obuku koje se sastoje od nekoliko različitih vrsta životinja. Ako korisnik unese novu sliku životinje koja se nalazi u skupini fotografija na kojoj model uči, algoritam će moći predvidjeti ispravnu životinju u tom skupu s određenom točnošću. Međutim, kada korisnik unese sliku životinje koja se ne nalazi u skupini fotografija na kojoj model uči, algoritam neće moći predvidjeti koja je to životinja te ju ne može smjestiti u neku od unaprijed određenih skupina.



Slika 3: Binarna klasifikacija (Ferre R.M, Fuente A., Lohan E.S.; 2019, izvor: https://www.researchgate.net/figure/SVM-binary-classification_fig2_33709684)

Slika predstavlja binarnu klasifikaciju koja se izvodi s dvije značajke. Značajke su predstavljene kao pozitivna i negativna klasa. Zato što je naš skup podataka dvodimenzionalan, svaka značajka ima određenu visinu i širinu. Korištenjem algoritma strojnog učenja s nadzorom naučeno pravilo jest: isprekidana linija označava granicu odlučivanja koja odvaja dvije značajke i klasificira ih u jednu od dviju kategorija s obzirom na njihove atribute i vrijednosti (javaTpoint, 2021).

3.4.1.2. Regresija

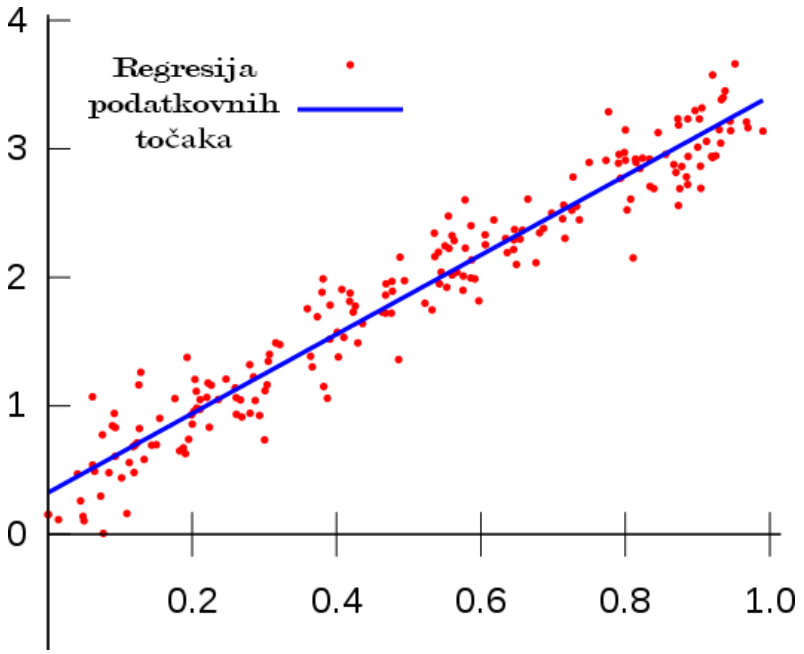
Sljedeća potkategorija strojnog učenja s nadzorom je regresija odnosno regresijska analiza. Prilikom korištenja regresijske analize, dan je skup varijabli prediktora i varijabli kontinuiranog odgovora, a cilj algoritma je pronaći odnos između varijabli koje omogućuju predviđanje rješenja (Gandhi R., 2018).

Drugim riječima, regresija uveliko pomaže razumjeti kako se određena vrijednost ovisne varijable mijenja prema neovisnoj varijabli, dok su ostale neovisne varijable fiksne.

Pomoću regresijske analize predviđaju se kontinuirane vrijednosti poput temperature, plaće, cijene itd. (javaTpoint, 2021).

Sarangam A. nalaže kako postoji nekoliko vrsta regresije, a opisane su neke od njih:

1. Linearna regresija – (eng. linear regression) je vrsta modela gdje se pretpostavlja da je odnos između nezavisne varijable i zavisne varijable linearan.
2. Logistička regresija – (eng. logistic regression) se koristi za procjenu vrijednosti određenih događaja koji se međusobno isključuju, na primjer, sretan/tužan.
3. Polinomska regresija – (eng. polynomial regression) je vrsta regresije koja modelira odnos vrijednosti zavisne varijable „x“ i nezavisne varijable „y“ kao nelinearan.
4. Regresija vektorske podrške – (eng. support vector regression) se može koristiti za rješavanje linearnih i nelinearnih modela.
5. Regresija stabla odluka – (eng. decision trees regression) se koristi za uklapanje sinusne krivulje s dodatnim šumnim promatranjem.
6. Regresija slučajne šume – (eng. random forest regression) je meta procjenitelj koji uklapa niz klasificirajućih stabala odlučivanja na različite poduzorke skupa podataka i koristi usrednjavanje za poboljšanje prediktivne točnosti.
7. Regresija grebena – (eng. ridge regression) se primjenjuje kada su nezavisne varijable visoko korelirane.
8. Lasso regresija – (eng. lasso regression) se koristi za dobivanje podskupa prediktora koji minimiziraju pogrešku predviđanja za varijablu kvantitativnog odgovora.



Slika 4: Linearna regresija (Fumo D.; 2018, izvor: https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06)

3.4.2. Učenje bez nadzora

Prilikom korištenja modela strojnog učenja bez nadzora, koriste se algoritmi strojnog učenja za analizu i grupiranje neoznačenih skupova podataka. Ono radi tako da algoritmi otkrivaju skrivene obrasce ili grupirane podatke bez ljudske pomoći. Zbog navedene sposobnosti otkrivanja sličnosti i razlika u skupini podataka, algoritam je odlično rješenje kod zadataka poput analize podataka, segmentacije kupaca i prepoznavanja slika (Alpaydin E., 2021).

Ovaj način strojnog učenja koristi se kod tri metode, a one su sljedeće: grupiranje, povezivanje i smanjenje dimenzionalnosti. U nastavku ćemo definirati svaku metodu, istaknuti uobičajene algoritme i pristupe za njihovo optimalno provođenje.

3.4.2.1. Grupiranje

Grupiranje je metoda podjele podataka u nekoliko grupa tako da su podaci u istim grupama slični podacima iz iste grupe, ali različiti od podataka iz neke druge grupe. To je u osnovi zbirka objekata na temelju sličnosti i različitosti među njima (Raschka S., 2017.).

Ekskluzivno grupiranje je model grupiranja koji zahtjeva da podatkovna točka može postojati samo u jednom klasteru. Ovo se također naziva i „tvrdo“ grupiranje. Preklapajuće grupiranje, za razliku od ekskluzivnog, dopušta da podatkovne točke postoje u više klastera s

različitim stupnjevima članstva. Ono se još naziva i „meko“ grupiranje (Shalev-Shwartz S., 2014).

Hijerarhijsko grupiranje je model algoritma za grupiranje bez nadzora koji se može svrstati u dvije skupine: aglomerativni i razdjelni. Ono radi tako da su u početku podatkovne točke izolirane poput zasebne grupe te se iterativno spajaju na temelju sličnosti sve dok se ne postigne cjelokupna skupina. Za određivanje sličnosti koriste se različite metode:

1. Wardova povezanost je metoda koja govori da je udaljenost dvaju klastera definirana povećanjem zbroja kvadrata nakon spajanja klastera.
2. Prosječna povezanost je metoda definirana srednjom udaljenošću dviju točaka u svakoj skupini.
3. Potpuna povezanost je metoda definirana najvećom udaljenošću između dvije točke u svakoj skupini.
4. Pojedinačna povezanost, za razliku od potpune, definirana je minimalnom udaljenošću između dvije točke u svakoj skupini

Razdjelno grupiranje definira se kao suprotnost aglomerativnom grupiranju te ono ima pristup „odozgo prema dolje“. To znači da se jedna skupina podataka dijeli na temelju razlika između podatkovnih točaka. Iako se ono uobičajeno ne koristi, potrebno ga je spomenuti. Postupci razdjelnog grupiranja obično se vizualiziraju pomoću dendrograma, dijagrama nalik stablu koji dokumentira spajanje ili razdvajanje podatkovnih točaka prilikom svake iteracije (Nielsen M., 2021).

3.4.2.2. Povezivanje

Metoda povezivanja se temelji na pravilima za pronalaženje odnosa između varijabli u danom skupu podataka. Ova se metoda strojnog učenja uveliko koristi u analizi tržišne korpe kako bi tvrtke bolje razumjele odnose između različitih proizvoda. Razumijevanje potrošačkih navika kupaca tvrtkama omogućuje razvoj strategija unakrsne prodaje i pokretača preporuka (Shalev-Shwartz S., 2014).

Primjer ove metode možemo vidjeti na većini online trgovina poput eBaya ili Amazona sa svojim „Kupci koji su kupili proizvod X, također su kupili i sljedeće proizvode“ ili pak sa Spotifyem i „Discover Weekly“ popisom pjesama.

Najpoznatiji algoritmi koji se koriste kod metode povezivanja su Apriori, Eclat i FP-Growth algoritmi (Nikolaiev, 2022).

Apriori je algoritam za učestalost rudarenja skupova podataka i učenja pravila pridruživanja pomoću relacijskih baza podataka. Radi tako da se identificiraju česte

pojedinačne stavke u bazi podataka te se proširuju na sve veće i veće skupove podataka sve dok se početni skupovi podataka pojavljuju dovoljno često u bazi podataka (IBM, 2020).

Eclat algoritam označava Equivalence Class Clustering i bottom-up Lattice Traversal. To je učinkovitija i skalabilnija verzija Apriori algoritma. Dok Apriori algoritam radi u horizontalnom smislu oponašajući pretraživanje grafa prvo u širinu, algoritam Eclat radi okomito te pretražuje graf u dubinu. Upravo taj okomit pristup pretraživanja čini Eclat algoritam bržim od Apriori algoritma (Korstanje J., The eclat algorithm, 2021).

FP-Growth algoritam ima cilj pronaći česte skupove stavke u skupu podataka, a da pritom bude brži od Apriori algoritma. Apriori algoritam u osnovi ide naprijed-natrag do skupa podataka kako bi provjerio postoji li istovremeno pojavljivanje proizvoda u skupu podataka. Kako bi FP-Growth algoritam bio brži, organizacija podataka mijenja se u stablo, a ne u skupove kao što je slučaj kod Apriori algoritma. Ovakva struktura podataka omogućuje brže skeniranje i ovdje algoritam dobiva na vremenu (Korstanje J., THE FP Growth algorithm, 2021).

3.4.2.3. Smanjenje dimenzionalnosti

Do sada smo pričali o metodama koje zahtijevaju što više podataka, no takav pristup može znatno utjecati na performanse algoritama strojnog učenja te isto tako otežati vizualizaciju skupova podataka. Smanjenje dimenzionalnosti je metoda koja se koristi kada je broj značajki ili dimenzija u danom skupu podataka prevelik. Glavni cilj ove metode jest smanjiti broj unosa podataka na određenu, uporabljivu količinu i istovremeno očuvati integritet skupa podataka što je više moguće. Obično se koristi u početnoj fazi, tijekom pretprocesiranja podataka (IBM, 2020).

Postoje dvije glavne kategorije smanjenja dimenzionalnosti, a to su odabir značajki i izdvajanje značajki.

Odabir značajki označava pohlepne algoritme koji se koriste za smanjenje dimenzionalnog prostora značajki danog skupa podataka. Cilj je dobiti sposoban model za automatski odabir podskupa značajki najrelevantnijih za problem s kojim se suočava u tom trenutku. Na taj se način poboljšava računalna učinkovitost procesa obuke, smanjuje šum skupa podataka, izbjegava prekomjerno prilagođavanje te se svime time zajedno smanjuje složenost modela. Izdvajanje značajki također smanjuje broj značajki određenog skupa podataka, ali za razliku od odabira značajki, značajke neće ostati iste kao u početku, odnosno prije korištenja ove metode. Prilikom korištenja metode izdvajanja značajki, svi podaci se premještaju u novi prostor značajki. U novom se prostoru nove značajke kombiniraju s prvotnima te one sadržavaju samo najrelevantnije podatke (IBM, 2020).

Dobro je poznato da algoritmi strojnog učenja zahtijevaju veliku količinu podataka kako bi naučili nepromjenljivost, obrasce i reprezentacije. Ako ti podaci sadrže veliki broj značajki, to može dovesti do problema jer kod procjene proizvoljnosti funkcija s određenom točnošću, broj značajki ili dimenzionalnosti potrebnih za procjenu eksponencijalno raste. To je osobito istinito s velikim podacima koji donose veću oskudnost (Nilesh B., 2022).

Rijetkost podataka obično se naziva značajkama koje imaju vrijednost nula, ali to ne znači da vrijednost nedostaje. Kada podaci imaju mnogo rijetkih značajki, prostor i računalna složenost znatno rastu. Tijekom treninga model uči buku i nije u mogućnosti dobro generalizirati podatke.

Kada su podaci rijetki, opažanja i uzorke u skupu podataka za obuku je teško grupirati jer visokodimenzionalni podaci uzrokuju da se svako opažanje u skupu podataka čini jednako udaljenim jedno od drugog. Ako su podaci smisleni i nisu redundantni, postojat će regije u kojima se slične podatkovne točke spajaju i grupiraju (Nilesh B., 2022).

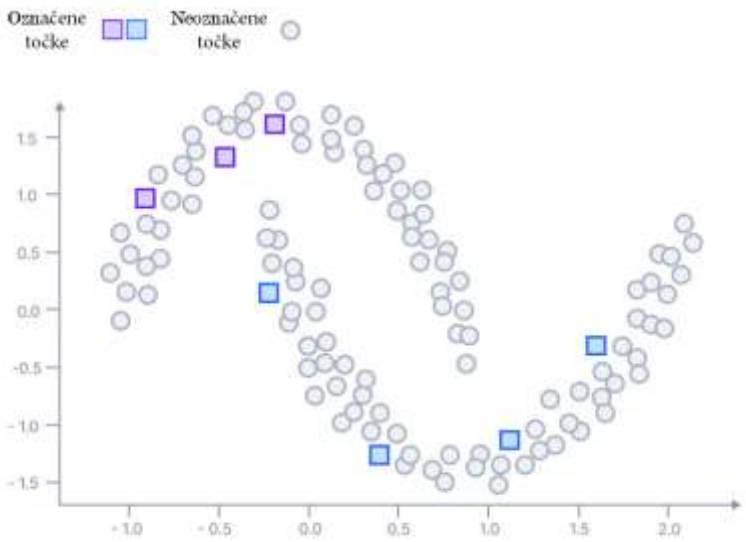
3.4.3. Polu-nadzirano učenje

Polu-nadzirano učenje vrsta je algoritma strojnog učenja koje se nalazi između učenja s nadzorom i strojnog učenja bez nadzora. Predstavlja međuosnovu između algoritama učenja s nadzorom i učenja bez nadzora te koristi kombinaciju označenih i neoznačenih skupova podataka tijekom razdoblja obuke (javaTpoint, 2021).

Glavna ideja polu-nadziranog učenja je različito postupanje s podatkovnom točkom na temelju toga ima li ona oznaku ili ne. Za označene točke algoritam će koristiti tradicionalni nadzor za ažuriranje težina modela. Kada je podatkovna točka neoznačena, algoritam minimizira razliku u predviđanjima između drugih sličnih primjera obuke (Bewtra A., 2022).

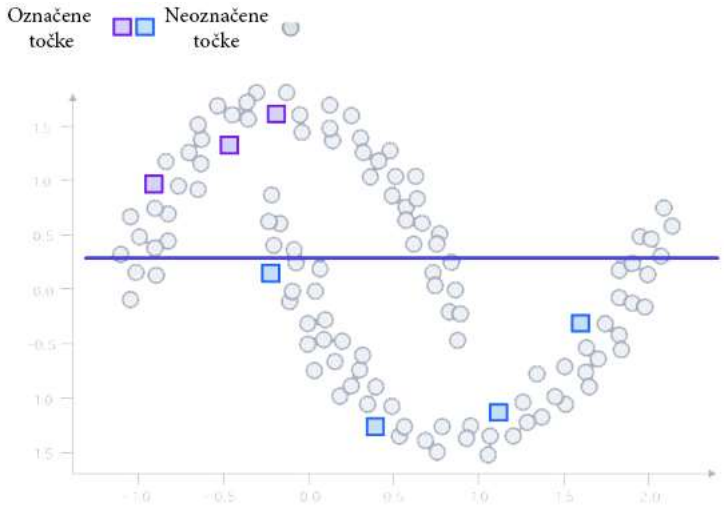
Kako bi se prevladali nedostaci učenja s nadzorom i algoritama za učenje bez nadzora, uvodi se koncept polu-nadziranog učenja. Njegov glavni cilj je učinkovito koristiti sve dostupne podatke, a ne samo označene podatke kao što je to slučaj kod učenja s nadzorom. U početku se slični podaci grupiraju zajedno s algoritmom učenja bez nadzora, a kasnije pomaže u označavanju neoznačenih podataka u označene podatke. Razlog tome je što su označeni podaci relativno skuplji od neoznačenih podataka (javaTpoint, 2021).

Kako bismo bolje razumjeli kako ovo učenje funkcionira, razmotrimo skup podataka o mjesecima na slici 5: problem binarne klasifikacije s jednom klasom za svaki polumjesec. Recimo da imamo 8 označenih podatkovnih točaka dok su sve ostale neoznačene.



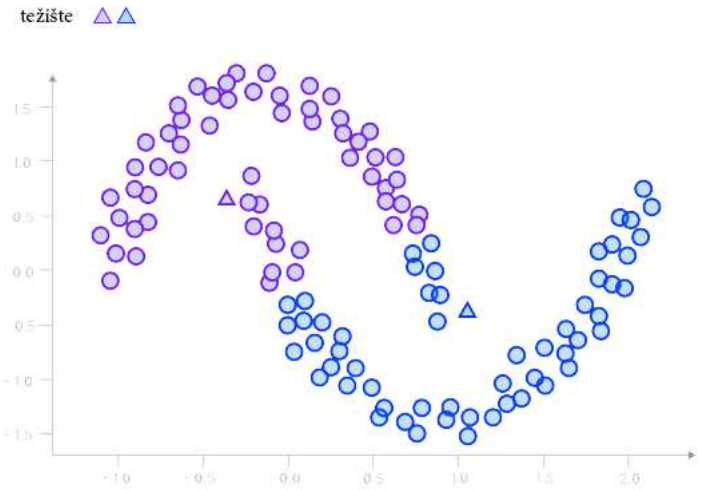
Slika 5: Binarna klasifikacija skupa podataka o mjesecima (Bewtra A.; 2022; izvor: https://www.v7labs.com/blog/semi-supervised-learning-guide)

Učenje s nadzorom ažurira težine modela kako bi se smanjila prosječna razlika između predviđanja i oznaka. Međutim, s ograničenim označenim podacima, ovo bi moglo pronaći granicu odluke koja vrijedi za označene točke, ali se neće generalizirati na cijelu distribuciju (Bewtra A., 2022).



Slika 6: Primjer učenja s nadzorom i granica odluke (Bewtra A.; 2022; izvor: https://www.v7labs.com/blog/semi-supervised-learning-guide)

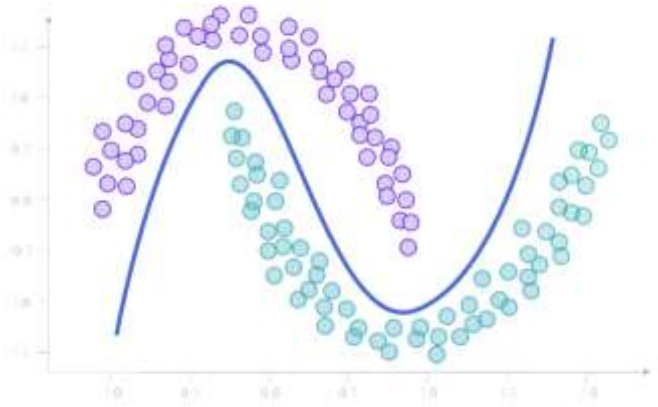
Učenje bez nadzora pokušava grupirati točke na temelju sličnosti u nekom prostoru značajki. Bez oznaka za usmjeravanje obuke, algoritam učenja bez nadzora mogao bi pronaći klastere koji nisu optimalni. Upravo tu situaciju možemo vidjeti na slici 7, gdje otkriveni klasteri netočno odgovaraju pravoj distribuciji klasa.



Slika 7: Grupiranje bez nadzora (Bewtra A.; 2022; izvor: https://www.v7labs.com/blog/semi-supervised-learning-guide)

Bez dovoljnog broja označenih podataka ili u teškim postavkama klasteriranja, učenje s nadzorom i učenje bez nadzora ne mogu uspjeti postići željeni rezultat. Međutim, kako je već navedeno, polu-nadzirano učenje koristi i označene i neoznačene podatke. Označene točke služe kao provjera razuma, odnosno, temelje naša predviđanja modela i dodaju strukturu problemu učenja utvrđujući koliko klasa postoji i koji klasteri odgovaraju kojoj klasi. Neoznačene podatkovne točke pružaju kontekst, drugim riječima, izlažući model što većem broju podataka, možemo točnije procijeniti oblik cijele distribucije. Tako da s oba dijela možemo trenirati preciznije i otpornije modele (Bewtra A., 2022).

U skupu podataka o mjesecima, polu-nadzirano učenje može se približiti istinskoj distribuciji prikazanoj na sljedećoj slici.



Slika 8: Istinska distribucija podatkovnih točaka (Bewtra A.; 2022; izvor: https://www.v7labs.com/blog/semi-supervised-learning-guide)

Polu-nadzirano učenje je vrlo moćno kada postoji ograničeni broj označenih podataka, a neoznačenih podataka je mnogo. U tom slučaju, model postaje izložen instancama na koje bi mogao naići u implementaciji, bez ulaganja vremena i novca u označavanje tisuća i tisuća dodatnih slika (Bewtra A., 2022) .

Glavne prednosti polu-nadziranog učenja su jednostavnost, laka razumljivost i visoka učinkovitost. Koristi se za rješavanje nedostataka algoritama učenja s nadzorom i učenja bez nadzora. Nedostaci ove vrste strojnog učenja su nestabilnost ponavljanja rezultata, niska točnost i nemogućnost primjene algoritama na podatke na razini mreže (javaTpoint, 2021).

3.4.4. Učenje s pojačanjem

„Glavni cilj strojnog učenja s pojačanjem je osposobljavanje modela za donošenje niza odluka u neizvjesnom, potencijalno složenom okruženju “ (Heidenreich H., 2018). Ovu vrstu učenja mogli bismo usporediti s nekom vrstom igre. Računalo koristi pokušaje i pogreške kako bi došlo do rješenja problema.

Da bi stroj uspješno odradio zadatke, umjetna inteligencija dobiva nagradu ako je radnja koju je izvela dobra, odnosno ispravna. Kada radnja nije ispravna, umjetna inteligencija dobiva kaznu. Cilj umjetne inteligencije je imati što veću nagradu (javaTpoint, 2021).

Uloga programera u ovoj vrsti učenja jest postavljanje pravila igre, odnosno kakva je politika nagrađivanja. Programer ne daje nikakve natuknice i prijedloge kako doći do rješenja već model sam mora otkriti put do rješenja. U početku model dolazi do nagrada slučajnim pokušajem, pa tako pomalo stvara sofisticiranu taktiku i nadljudske vještine. „Zbog velikog broja pretraživanja i ispitivanja, učenje s pojačanjem trenutno je najučinkovitiji način pokazivanja kreativnosti stroja“ (Alpaydin E., 2021). Ono što ga čini superiornijim od ljudi jest

to što umjetna inteligencija prikuplja iskustva iz stotine paralelnih igranja u istom vremenu u kojem ljudska osobe ima jedno igranje. Naravno, broj i intenzitet igranja ovisi o jačini računalne infrastrukture. Iz toga razloga, ova se vrsta strojnog učenja počela intenzivnije razvijati 1990-tih godina. Danas je razvoj puno brži zbog razvoja novih moćnih računalnih tehnologija koje otvaraju vrata novim, još netaknutim problemima koje računala u prošlosti nisu mogla obavljati.

Jedan od najpoznatijih takvih primjera bio bi obuka modela koji upravljaju autonomnim automobilima. Kod ovog primjera, glavni cilj programa bio bi postaviti sigurnost putnika i ljudi oko vozila na prvo mjesto, poštivanje znakova, smanjenje vremena putovanja i zagađenja. Kada je riječ o autonomnom trkaćem automobilu, brzina vozila bi bila prioritet. U takvim slučajevima programer ne može znati što će se i što se može dogoditi na cesti te umjesto da piše tisuće instrukcija, programer priprema pomoćno sredstvo za učenje kako bi bilo sposobno učiti iz sustava nagrada i kazni (Heidenreich H., 2018).

U tipičnim slučajevima učenja s pojačanjem, poput pronalaska najkraćeg puta između dvije točke na karti, rješenje nije apsolutna vrijednost. Rješenje poprima ocjenu učinkovitosti izraženu u postotku. Što je veći postotak, algoritam dobiva bolju nagradu.

3.4.5. Razlike u vrstama učenja

Algoritmi učenja s nadzorom koriste označene podatke, za razliku od algoritama bez nadzora koji koriste neoznačene skupove podataka. Strojno učenje s nadzorom pomoću označenih podataka predviđa ishode u budućnosti ili dodjeljuje podatke određenoj kategoriji. Kategorija se određuje ovisno o problemu koji se pokušava riješiti, odnosno na temelju problema regresije ili klasifikacije. Strojno učenje s nadzorom je točnije od učenja bez nadzora, ali ono zahtjeva ljudsku pomoć kako bi podaci biti pravilno označeni. S druge strane, učenje s nadzorom izbjegava računalnu složenost jer nije potrebna velika i složena obuka za dolaženje do željenih rezultata. Što se tiče izlaznih podataka, kod učenja s nadzorom, podaci su poznati sustavu, dok se kod učenja bez nadzora izlazni podaci temelje na prikupljanju percepcija o podacima. Učenje s pojačanjem ima agenta, pomoćnika, koji komunicira s okolinom u diskretnim koracima. Nadgledanost u učenju s pojačanjem je između prve dvije tehnike, ali ono pretežno ovisi o agentu u određivanju. Učenje s nadzorom predviđa ishode putem podataka u klasi, odnosno tipu klase. Učenje bez nadzora otkriva značajke unutar podatkovnih točaka, dok učenje s pojačanjem uči putem sistema nagrade i kazne.

Algoritmi učenja s nadzorom obično su točniji od modela učenja bez nadzora, no oni zahtijevaju prethodnu ljudsku intervenciju kako bi podaci bili ispravno označeni. Označeni skupovi podataka također dopuštaju učenju s nadzorom da izbjegnu računsku složenost zato

što nije potreban veliki skup za treniranje kako bi došao do željenih rezultata. Polu-nadzirano učenje događa se kada je samo dio zadanih ulaznih podataka označen. Učenje bez nadzora i polu-nadzirano učenje može biti privlačnija alternativa jer učenje s nadzorom može biti dugotrajan i skup proces zbog oslanjanja na stručnost ljudi prilikom testiranja i označavanja podataka na odgovarajući način (IBM, 2020).

3.5. Neuronske mreže

Neuronske mreže, poznate i kao umjetne neuronske mreže podskup su strojnog učenja i čine srce algoritama dubokog učenja. Umjetne neuronske mreže sastoje se od slojeva čvorova koji sadrže ulazni sloj, jedan ili više skrivenih slojeva te na kraju izlazni sloj. Svaki se čvor, odnosno neuron povezuje s drugim i sadrži povezanu težinu i prag. Ako je izlaz nekog čvora iznad navedenog praga, isti se aktivira i šalje podatke na sljedeći sloj neuronske mreže. U suprotnom, podaci se ne prenose na sljedeći sloj mreže (Nielsen M., 2021).

Neuronska se mreža oslanja na podatke o treniranju kako bi s vremenom poboljšali svoju performansu i točnost. Nakon što su oni dospjeli do određene preciznosti i točnosti, postaju moćni alat u računalnoj znanosti i umjetnoj inteligenciji. Precizne neuronske mreže dopuštaju nam klasificiranje i grupiranje podataka velikom brzinom. Kada je riječ o zadacima s prepoznavanjem govora ili slike, oni mogu trajati nekoliko minuta, dok bi ljudskoj osobi za isti zadatak bili potrebni sati. Najpoznatija neuronska mreža je Googleov algoritam pretraživanja (IBM, 2022).

3.5.1. Vrste neuronskih mreža

Neuronske mreže mogu se klasificirati u različite vrste koje se koriste prilikom rješavanja različitih problema. Iako ovo nije sveobuhvatan popis vrsta neuronskih mreža, predstavljene su najčešće vrste na koje se nailazi zbog njihovog čestog korištenja.

Perceptron

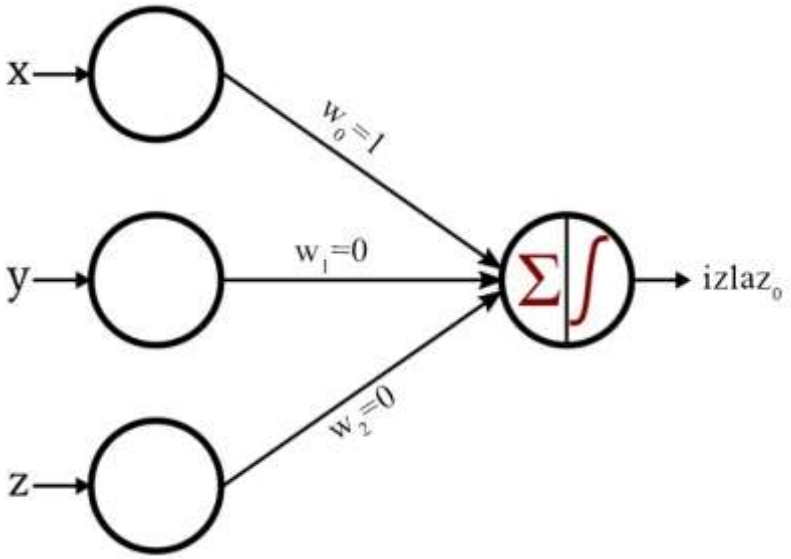
Osnovna gradivna jedinica umjetnih neuronskih mreža je perceptron. Perceptron je stvoren 1958. godine od strane znanstvenika Franka Rosenblatta, što ju čini najstarijom vrstom neuronske mreže. On je najjednostavniji tip neuronske mreže jer je on ustvari binarni klasifikacijski algoritam. Sastoji se od jednog neurona koji uzima niz podataka kao ulaz i pritom predviđa oznaku klase. To je moguće zbog izračunavanja težinskog zbroja ulaza i pristranosti. Težinski zbroj ulaznih podataka naziva se aktivacija (Nielsen M., 2021).

$$Aktivacija = Težine * ulazi + pristranost$$

Postoje dvije vrste izlaza, on može biti 0 ili 1. Slučajevi su sljedeći:

$$izlaz = 1 : Ako\ je\ aktivacija > 0$$

$$izlaz = 0 : Ako\ je\ aktivacija \leq 0$$



Slika 9: Perceptron (Keim R.; 2019; izvor: https://www.allaboutcircuits.com/technical-articles/how-to-train-a-basic-perceptron-neural-network/)

Napredna umjetna neuronska mreža

Glavna značajka napredne neuronske mreže jest da je ona umjetna neuronska mreža u kojoj veze između čvorova ne tvore ciklus. Napredna umjetna neuronska mreža je najvažniji model dubokog učenja, a glavni cilj joj je odrediti neku funkciju *f**. Na primjer, klasifikator, *y=f * (x)* preslikava ulaz *x* u kategoriju *y*. Napredna neuronska mreža definira preslikavanje *y=f (x; θ)* te uči vrijednosti parametra θ čime se dolazi do najbolje aproksimacije funkcije (Shalev-Shwartz S., 2014).

Ovaj model se naziva napredni jer informacije teku kroz funkciju koja se evaluira iz *x*, kroz međuproračune koji se koriste za definiranje funkcije *f* i tako sve do izlaza *y*. Ne postoje povratne veze u kojima bi se izlazi mogli vraćati u sebe. Kada bi se uključile povratne veze, tada bismo govorili o ponavljajućim neuronskim mrežama.

Ponavljajuće neuronske mreže

Ponavljajuća neuronska mreža vrsta je umjetne neuronske mreže koja koristi sekvencijalne podatke ili podatke vremenskih serija. Upotreba ovih neuronskih mreža pronašla se kod rješavanja problema poput prijevoda jezika, obrade jezika i prepoznavanje govora. Ponavljajuća neuronska mreža se kao i konvolucijske mreže koristi podacima za obuku učenja. Neuronske mreže uzimaju informacije iz prethodnih ulaza kako bi utjecali na trenutni ulaz i izlaz. Tradicionalne neuronske mreže pretpostavljaju da su ulaz i izlaz neovisni jedan o drugom, izlaz ponavljajuće neuronske mreže ovisi o prethodnim elementima unutar niza (Nielsen M., 2021).

Iako bi budući događaji mogli biti korisni prilikom određivanja izlaza danog niza, jednosmjerne ponavljajuće neuronske mreže ne mogu uzeti u obzir buduće događaje u svojim predviđanjima.

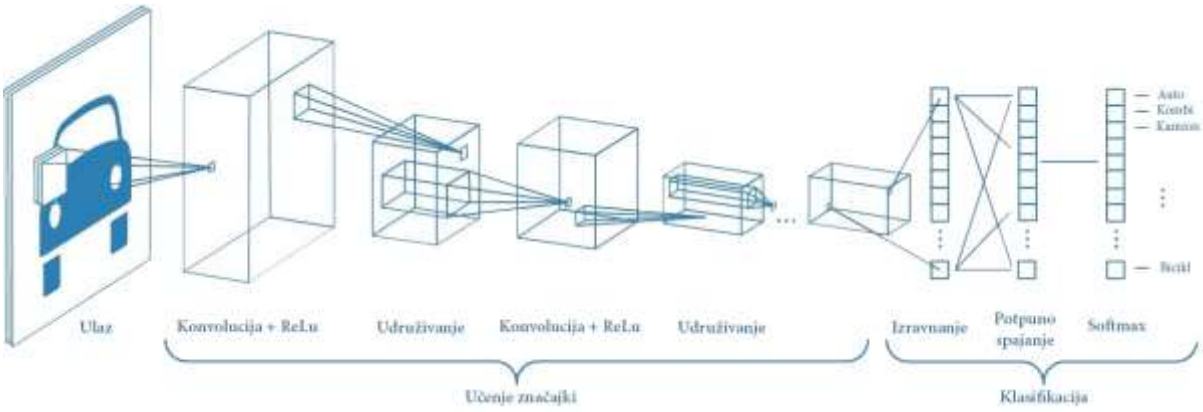
Konvolucijske neuronske mreže

Konvolucijska neuronska mreža algoritam je dubokog strojnog učenja koje može uzeti ulaznu sliku, dodijeliti određenu važnost različitim objektima na slici i razlikovati ih jedne od drugih. Ljudska intervencija u početku obrade podataka puno je niža u usporedbi s ostalim algoritmima za klasifikaciju. Kada je riječ o ostalim algoritmima za klasifikaciju, filtri su rađeni ručno, odnosno korisnik sam unosi određene karakteristike za prepoznavanje objekta sa slike. S druge strane, konvolucijska mreža uz dovoljno obuke ima sposobnost sama naučiti te filtere, odnosno karakteristike. Ova mreža sastoji se od tri vrste slojeva, a to su: konvolucijski sloj, skupni sloj, potpuno povezani sloj (Nielsen M., 2021).

Konvolucijski se slojevi mogu dodavati jedan na drugoga te sa svakim novim slojem konvolucijska mreža postaje sve složenija, identificirajući veće dijelove slike. Prvi slojevi zaduženi su na jednostavne značajke poput boje ili rubova. Što je više rubova, tako se počinju prepoznavati veći elementi ili oblici objekata sve dok se ne identificira željeni objekt. Bitno je napomenuti da je detektor značajki dvodimenzionalni niz koji predstavlja dio slike. Iako se razlikuju po veličini, veličina filtera je obično 3x3 matrica. Filter se primjenjuje na određeni dio slike, a točkasti se proizvod izračunava između ulaznih piksela i filtera te se on unosi u izlazni niz. Filter obavlja jedan te isti posao sve dok ne prođe cijelu sliku. Konačni izlaz iz niza proizvoda s točkama iz ulaza i filtera poznati je kao karta značajki (Shalev-Shwartz S., 2014).

Skupni sloj provodi smanjenje dimenzionalnosti čime se smanjuje broj parametara u ulazu. Isto kao i kod konvolucijskog sloja, koristi se filter sve dok se ne prođe cijelom slikom, ali razlika je što filter kod skupnog sloja nema nikakvu težinu. Iz tog razloga, jezgra je zadužena za funkciju agregacije na vrijednosti unutar prijemnog polja, popunjavajući izlazni niz. U ovom

sloju postoje dvije vrste udruživanja. Maksimalno udruživanje radi na način da kako se filter pomiče po ulazu, on odabire piksel s najvećom vrijednošću za slanje u izlazni niz. Prosječno udruživanje radi na sljedećem principu: kako se filter pomiče po ulazu, on izračunava prosječnu vrijednost unutar prijemnog polja za slanje u izlazni niz. U ovom se sloju gubi mnogo podataka, ali konvolucijska neuronska mreža svejedno donosi brojne prednosti, a neke od njih su smanjenje složenosti, poboljšavanje učinkovitosti i ograničenje rizika prekomjernog opremanja (Nielsen M., 2021).



Slika 10: Primjer konvolucijske neuronske mreže (Saha S.; 2018; izvor: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)

3.6. TensorFlow

„TensorFlow[3] je softverska biblioteka otvorenog kôda za numeričko izračunavanje pomoću grafikona protoka podataka“ (TensorFlow, 2022). Ona se temelji na protoku podataka i različitom programiranju, ali ima poseban fokus na obuku i upravljanje s dubokim neuronskim mrežama.

Razvijen je od strane Google Brain tima za unutarnje poslove, ali 2015. dostupna je i za širu javnost s podrškom za Python,C++, JavaScript, Javu i ostale programske jezike (Pedamkar P., 2022).

Koristi se za rješavanje različitih zadataka poput prepoznavanja elemenata na slici, obradu teksta i predviđanje.

TensorFlow je jedna od najkorištenijih softverskih biblioteka, a neke od svjetski poznatih tvrtki koje koriste njegove mogućnosti su: Twitter, Airbnb, Uber itd. Najpoznatije korištenje je

[3] https://www.tensorflow.org/

naravno kod Google proizvoda poput Gmail-a, prevoditelja i same tražilice. Iako ima podršku za Python, C++, Javu i ostale programske jezike, TensorFlow koristi programski jezik Python za sučelje između korisnika i pozadinskog računanja zbog svoje jednostavnosti i popularnosti u posljednjim godinama. Pozadinsko računanje je ipak napisano u C++ jer je on puno brži i bolje optimizirani od programskog jezika Python (Yegulalp S., 2022).

Iako se TensorFlow koristi u mnogim složenim aplikacijama, rad s bibliotekom je vrlo jednostavan i lagan. To možemo vidjeti i u samom početku jer već s dvadesetak linija kôda moguća je implementacija jednostavne neuronske mreže koja prepoznaje znakove uz pomoć MNIST baze podataka. MNIST je baza podataka ručno napisanih znamenki koja se koristi prilikom obuke sustava za obradu slika i strojnog učenja. Za ovakvu jednostavnost korištenja i izrade algoritama zaslužna je biblioteka Keras zbog koje korisnik, tj. programer ne mora znati sve detalje algoritama koje koristi, ni točnu arhitekturu neuronske mreže. TensorFlow biblioteka omogućuje korisniku da slaže vlastite slojeve neuronske mreže koja se kasnije prevodi.

```
1   import tensorflow as tf
2
3   mnist = tf.keras.datasets.mnist
4
5   (x_train, y_train), (x_test, y_test) = mnist.load_data()
6   x_train, x_test = x_train / 255.0, x_test / 255.0
7
8   model = tf.keras.models.Sequential([
9    tf.keras.layers.Flatten(input_shape=(28, 28)),
10   tf.keras.layers.Dense(128, activation='relu'),
11   tf.keras.layers.Dropout(0.2),
12   tf.keras.layers.Dense(10)
13  ])
14
15  predictions = model(x_train[:1]).numpy()
16  predictions
17
18  tf.nn.softmax(predictions).numpy()
```

Slika 11: Jednostavna neuronska mreža (TensorFlow; 2022; izvor: https://www.tensorflow.org/tutorials/quickstart/beginner)

Prvi korak prilikom korištenja biblioteke TensorFlow je njezino postavljanje, a način na koji se to radi je prikazan u liniji broj 1. U liniji 3 učitavamo MNIST skup podataka, a u liniji 5 i 6 podatke pripremamo za obradu tako da podatke iz cijelih brojeva pretvorimo u brojeve s pomičnim zarezom. Sljedeći korak je izgradnja *tf.keras* sekvencijalnog modela slaganjem slojeva. Za svaki primjer, model vraća vektor neobrađenih predviđanja, jedan za svaku klasu. Funkcija *tf.nn.softmax* pretvara te vektore neobrađenih predviđanja u vjerojatnosti za svaku klasu. Na sljedećoj slici moguće je vidjeti kako ta vjerojatnost izgleda.

```
array([[0.11960829, 0.06124588, 0.0764901 , 0.30181262, 0.08770514,
        0.15668967, 0.04416083, 0.05969675, 0.05006609, 0.04252464]],
      dtype=float32)
```

Slika 12: Prikaz vjerojatnosti u modelu (TensorFlow; 2022; izvor: https://www.tensorflow.org/tutorials/quickstart/beginner)

Slika 12 prikazuje vjerojatnosti za svaku klasu. Vidljivo je kako postoji 10 različitih kategorija koje se klasificiraju.

```
19    loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
20    loss_fn(y_train[:1], predictions).numpy()
21    model.compile(optimizer='adam',
22            loss=loss_fn,
22            metrics=['accuracy'])
23    model.fit(x_train, y_train, epochs=5)
24    model.evaluate(x_test,  y_test, verbose=2)
```

Slika 13: Testiranje i validacija (TensorFlow; 2022; izvor: https://www.tensorflow.org/tutorials/quickstart/beginner)

U liniji 19 definiramo funkciju gubitka za obuku koristeći *losses.SparseCategoricalCrossentropy* koji uzima vektor neobrađenih predviđanja i True indeks, pa vraća skalarni gubitak za svaki primjer. Prije početka obuke, konfiguriramo i kompajliramo model koristeći Keras model.compile. Postavljamo klasu optimizatora na *adam*, gubitak se postavlja na funkciji *loss_fn* koja je definirana u liniji 20. Na kraju se navode metrike koje će se procijeniti za model postavljanjem parametra metrike na točnost. Metodom *model.fit* upotrebljava se za prilagodbu parametara modela i minimiziranje gubitka. Na samom kraju, potrebno je provjeriti točnost kôda, a to se radi pomoću metode *model.evaluate*. Primjer preuzet sa stranice TensorFlow.

TensorFlow biblioteka sadrži dva alata na raspolaganju. TensorBoard je skup alata koji pruža vizualizaciju rezultata i olakšava korisniku praćenje različitih metrika kao što su točnost i gubitak zapisa na skupu za obuku ili validaciju. Drugi alat je TensorFlow Serving koji omogućuje implementaciju novih funkcionalnosti i algoritama u već kreiranom kôdu uz pomoć API-a (Raschka S., 2017.).

Slika 14: TensorFlow logo (TensorFlow github; 2022, izvor: https://github.com/tensorflow/docs)

Tensorflow omogućuje programerima stvaranje grafova struktura koje opisuju kako se podaci kreću kroz graf ili niz čvorova za obradu. Svaki čvor u grafu predstavlja matematičku operaciju, a svaka veza ili rub između čvorova je višedimenzionalni niz podataka ili tenzor. To znači da je programerima omogućeno stvaranje velikih neuronskih mreža s mnogo slojeva. Sljedeća odlična stvar vezana uz TensorFlow je da se aplikacije mogu pokrenuti u gotovo svim uređajima koji su pogodni: lokalno računalo, klaster u oblaku, iOS i Android, CPU ili GPU (Yegulalp S., 2022).

Ako koristimo Googleov vlastiti oblak, moguće je poslati vlastiti kôd modela izravno na vlastiti Google Cloud račun i koristiti Google Cloud računalne resurse bez potrebe za prijavom i interakcijom s Cloud korisničkim sučeljem (nakon postavljanja projekta na konzoli) (TensorFlow, 2022).

TensorFlow 2.0, objavljen u listopadu 2019., preinačio je okvir na mnogo načina na temelju povratnih informacija korisnika, kako bi ga učinio lakšim za rad (npr., korištenjem relativno jednostavnog API-ja Keras za obuku modela) i učinio učinkovitijim. Distribuiranu obuku lakše je izvoditi zahvaljujući novom API-ju, a podrška za TensorFlow Lite omogućuje implementaciju modela na više različitih platformi (Yegulalp S., 2022).

TensorFlow Lite je također softverska biblioteka otvorenog kôda, međuplatformski okvir dubokog učenja koji pretvara unaprijed obučeni model u TensorFlowu u poseban format koji se može optimizirati za brzinu ili pohranu. Ovaj model može se primijeniti na rubnim uređajima kao što su Android ili iOS mobiteli ili ugrađene uređaje temeljene na Linuxu poput Raspberry Pi ili mikro kontrolera (TensorFlow, n.d.).

Glavne komponente TensorFlowa

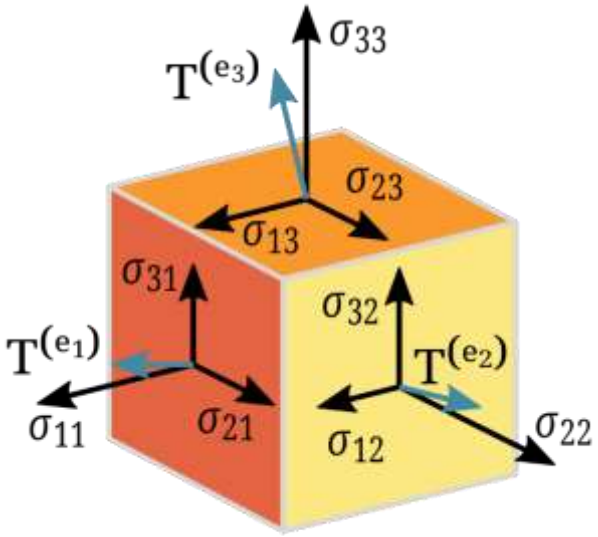
Do sada smo pričali općenito o TensorFlowu, a sada je potrebno spomenuti glavne komponente TensorFlowa. Tenzori su glavne komponente TensorFlowa. Oni su definirani kao osnovne strukture podataka tj. višedimenzionalni niz ili popis. Spojni rubovi u bilo kojem dijagramu toka su tenzori. To su višelinearne karte koje mogu biti bilo što, od vektorskih

prostora do realnih brojeva. Drugim riječima, tenzor može biti skalar, vektor ili matrica. On može biti rezultat izračuna ili može potjecati iz ulaznih podataka (Pedamkar P., 2022).

Tenzor se identificira pomoću sljedeća tri parametra:

- Poredak – jedinica dimenzionalnosti opisana unutar tenzora naziva se poredak, ona identificira broj dimenzija tenzora
- Oblik – broj redaka i stupaca zajedno definira oblik tenzora
- Tip – opisuje tip podataka dodijeljen tenzorovim elementima

Za izgradnju tenzora, potrebno je razmotriti izgradnju i pretvaranje n-dimenzionalnog niza. Moguće su dvije dimenzije, a to su sljedeće: jednodimenzionalni tenzor je normalna struktura niza koja uključuje jedan skup vrijednosti iste vrste podataka te dvodimenzionalni tenzor za čije se stvaranje koristi niz nizova (Pedamkar P., 2022).

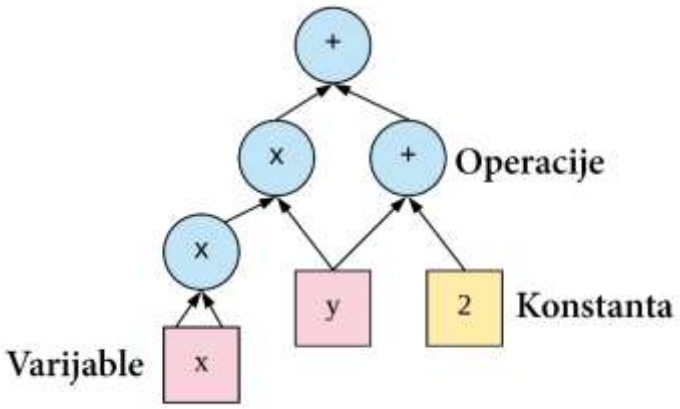


Slika 15: Tenzor (Wikipedia; 2022; izvor: https://en.wikipedia.org/wiki/Tensor#/media/File:Components_stress_tensor.svg)

Kada govorimo o glavnim komponentama, potrebno je spomenuti graf. Sve što se događa u modelu, predstavljeno je računskim grafikonom. To ga čini idealnim mjestom za sve što je povezano s modelom.

Grafovi su strukture podataka koje sadrže skup *tf.Operation* objekata, koji predstavljaju jedinice izračuna i *tf.Tensor* objekti, koji predstavljaju jedinice podataka koje teku između operacija. Oni su definirani u *tf.Graph* kontekstu. Budući da su ti grafikoni podatkovne strukture, mogu se spremati, pokretati i vraćati bez izvornog Python kôda (TensorFlow, 2022).

U osnovi, to znači da je graf raspored čvorova koji predstavljaju operacije u modelu. Najbolji način da to objasnimo bio bi preko jednostavnog primjera. Recimo da želimo napisati kôd za sljedeću funkciju: $f(x, y) = x^2y + y + 2$. Graf u TensorFlowu izgledat će na sljedeći način:



Slika 16: Shema računskog grafa u TensorFlowu (Easy-TensorFlow; 2018; izvor: https://www.easy-tensorflow.com/tf-tutorials/basics/graph-and-session)

Graf se sastoji od niza čvorova međusobno povezanih rubova. Možemo vidjeti kako nam x i y predstavljaju varijable, 2 je konstanta, a matematičke operacije su zbrajanje(+) i množenje(x). Prema grafu imamo jedan čvor za svaku operaciju, bilo za operacije na tenzorima (matematičke operacije) ili generiranje tenzora (varijable i konstante). Svaki čvor uzima nula ili više tenzora kao ulaze i proizvodi tenzor kao izlaz.

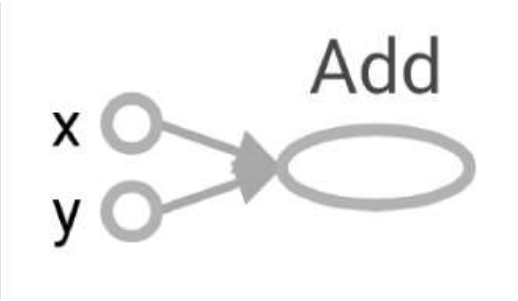
Sada ćemo na jednostavnijem primjeru prikazati kako izgleda operacija zbrajanja u TensorFlowu. Programski kôd zbrojit će dvije vrijednosti, recimo $a = 4$ i $b = 6$. Kako bi zbrajanje bilo moguće, potrebno je pozvati *tf.add()*. Funkcija *tf.add()* ima tri argumenta, a to su 'x', 'y' i 'name' gdje su x i y vrijednosti koje će se zbrojiti, a 'name' je naziv operacije.

```
import tensorflow as tf
a = 4
b = 6
zbroj = tf.add(a, b, name='Add')
print(zbroj)
```

```
tf.Tensor(10, shape=(), dtype=int32)
```

Slika 17: Prikaz operacije zbrajanja u TensorFlowu (Izrada autora, 2022.)

Korištenjem alata Tensorboard generira se sljedeći graf za gore navedeni kôd.



Slika 18: Generirani graf za zbrajanje (Izrada autora, 2022.)

Programski kôd kreirao je 2 ulazna čvora x i y (vrijednosti $a = 4, b = 6$) i izlazni čvor 'Add' za operaciju zbrajanja. Ovo je jednostavan primjer na kojem možemo vidjeti kako program ustvari radi. Kada bismo željeli rekreirati shemu grafa sa slike 8, potrebne bi bile funkcije *tf.multiply()* i *tf.pow().*

Zašto koristiti TensorFlow?

Najveća pojedinačna prednost koju TensorFlow pruža razvoju strojnog učenja je apstrakcija. Umjesto da se bavi sitnim detaljima implementacije algoritama ili smišljanjem ispravnih načina za spajanje izlaza jedne funkcije s ulazom druge, programer se može usredotočiti na cjelokupnu logiku aplikacije (Yegulalp S., 2022).

Možemo reći kako TensorFlow brine o različitim detaljima iza kulisa.

TensorFlow nudi dodatne pogodnosti za programere kada je riječ o otklanjanju pogrešaka i sticanju introspekcije u aplikacijama. Svaka operacija grafa može se procijeniti i modificirati zasebno i transparentno, umjesto da se cijeli graf konstruira kao jedan neproziran objekt i procjenjuje sve odjednom.

Vizualizacijski paket TensorBoard omogućuje pregled i profiliranje načina na koji se grafikoni izvode putem interaktivne nadzorne ploče temeljene na webu. Usluga Tensorboard.dev omogućuje učitavanje i dijeljenje eksperimenata strojnog učenja napisanih u TensorFlowu. Besplatna je za korištenje s pohranom do 10 milijuna skalara, 1GB tenzorskih podataka i 1GB podataka binarnih objekata (Yegulalp S., 2022).

Prednosti i nedostaci TensorFlowa

Nakon općenitosti TensorFlowa, njegovim komponentama i kratkom prikazu samog rada, potrebno je spomenuti i neke od prednosti TensorFlowa. Prednosti i nedostaci preuzeti su sa stranice (GeeksforGeeks, 2022).

Skalabilnost - TensorFlow nije ograničen na samo jedan određeni uređaj. Djeluje jednako učinkovito na mobilnom uređaju kao i na bilo kojem drugom složenom stroju. Knjižnica je tako definirana da njezina implementacija nije ograničena na jedan uređaj.

Platforma otvorenog kôda - dostupan je besplatno svima koji ga žele koristiti. Ova značajka omogućuje svakom korisniku korištenje ovog modula kad god i gdje god im je on potreban.

Graf - TensorFlow trenutno ima bolju moć vizualizacije podataka od bilo koje druge dostupne knjižnice. Ta značajka olakšava rad na neuronskim mrežama.

Otklanjanje pogrešaka - Tensorboard omogućuje jednostavno uklanjanje pogrešaka u čvorovima. To pomaže smanjiti troškove i vrijeme prolaženja kroz čitav kôd.

Paralelizam - TensorFlow za svoj rad koristi GPU i CPU sustave. Korisnik može slobodno koristiti bilo koju arhitekturu prema zahtjevu. Sustav koristi GPU ako nije drugačije određeno. Tim postupkom donekle se smanjuje korištenje memorije. Zbog toga se ovaj kapacitet TensorFlowa smatra kao biblioteka za hardversko ubrzanje.

Kompatibilan - kompatibilan je s mnogim programskim jezicima kao što su Python, C++, JavaScript, Swift, itd. To korisnicima omogućuje rad u okruženju u kojem se najbolje snalaze i koje im je ugodno.

Upravljanje knjižnicom - budući da je podržan od strane Googlea, TensorFlow se često ažurira i može pokazati izvanredne performanse.

Usprkos brojnim prednosti, treba spomenuti i nedostatke.

Slaba podrška za Windows - TensorFlow ima vrlo ograničen skup značajki za Windows korisnike. Za korisnike Linux operacijskog sustava to nije slučaj jer postoji širok niz značajki.

Neefikasnost - razmjerno je sporiji i manje upotrebljiv u usporedbi s konkurentskim okvirima.

GPU podrška - TensorFlow sadrži samo NVIDIA podršku za GPU i podršku programskog jezika Python za GPU programiranje.

Česta ažuriranja - iako smo ovo spomenuli kao prednost, ono je ujedno i nedostatak zato što korisnik s vremena na vrijeme mora reinstalirati TensorFlow kako bi se mogao vezati i pomiješati s najnovijim ažuriranjima.

Nedosljedan - TensorFlow sadrži homonime kao nazive svog sadržaja što korisniku otežava pamćenje i korištenje. Jedino ime se koristi za različite svrhe gdje se javlja problem i nastaje zbrka.

Ovisnost - iako TensorFlow smanjuje veličinu programa i čini ga jednostavnim za korištenje, dodaje mu sloj složenosti. Svaki kôd treba neku platformu za svoje izvršenje što povećava ovisnost.

Konačno smo vidjeli prednosti i nedostatke TensorFlowa. On pruža mnoge značajke koje ga čine prikladnim za izgradnju sustava temeljenih na strojnom i dubokom učenju, ali odbacuje neke sposobnosti kako bi ga učinile idealnom platformom. Također je važno napomenuti da su prednosti i nedostaci napisani i ažurirani u ljeti 2022. godine te je moguće da će neki od nedostataka biti ispravljeni, odnosno da ih neće biti.

4. Primjer izrade modela pomoću TensorFlowa

Dosad je riječ bila o teoriji i pravilnom korištenju određenih metoda strojnog učenja. Sada je red da tu teoriju upotrijebimo za izradu programskog kôda, odnosno dobivanje rješenja. Programski kôd objasnit ćemo u svakoj fazi, od prikupljanja podataka, implementacije, testiranja i na kraju prikaz rezultata testiranja.

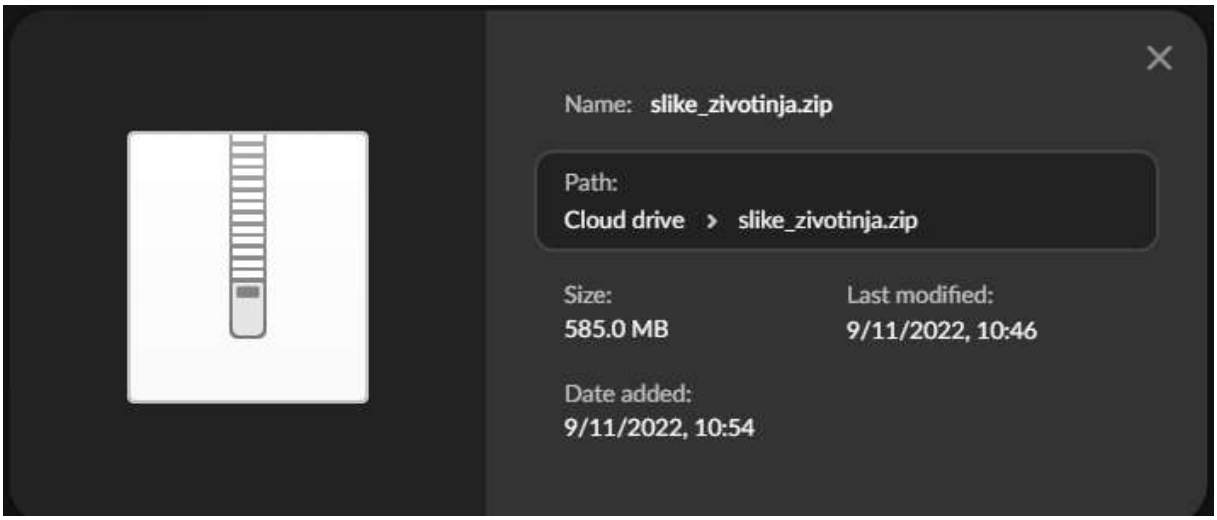
4.1. Prikupljanje podataka

Prvi korak svakog projekta strojnog učenja je prikupljanje kvalitetnih podataka za obradu. Za prikupljanje naših podataka, uveliko pomaže opcija „Google slike“ s kojih su slike životinja bile preuzete.

Životinja	Kokoši	Konji	Krave	Leptiri	Mačke	Ovce	Pauci	Psi	Slonovi	Leptiri	Ukupno
Broj slika	3098	2623	1866	2112	1668	1820	4821	4863	1446	1862	26179

Tablica 1: Broj slika po skupini životinja (Izrada autora, 2022.)

U suštini, raspolažemo s 26 179 slika različitih životinja, što zauzima otprilike 600MB. Slike također nisu u originalnoj rezoluciji već su u smanjenom što također pomaže u brzini izvođenja testiranja. Slike je moguće preuzeti sa sljedećeg linka: https://mega.nz/file/Zl5n0BrR#sWxN3kfbu4_LV-6C32nO1znFK3phINztvG-UdbkJsF0.



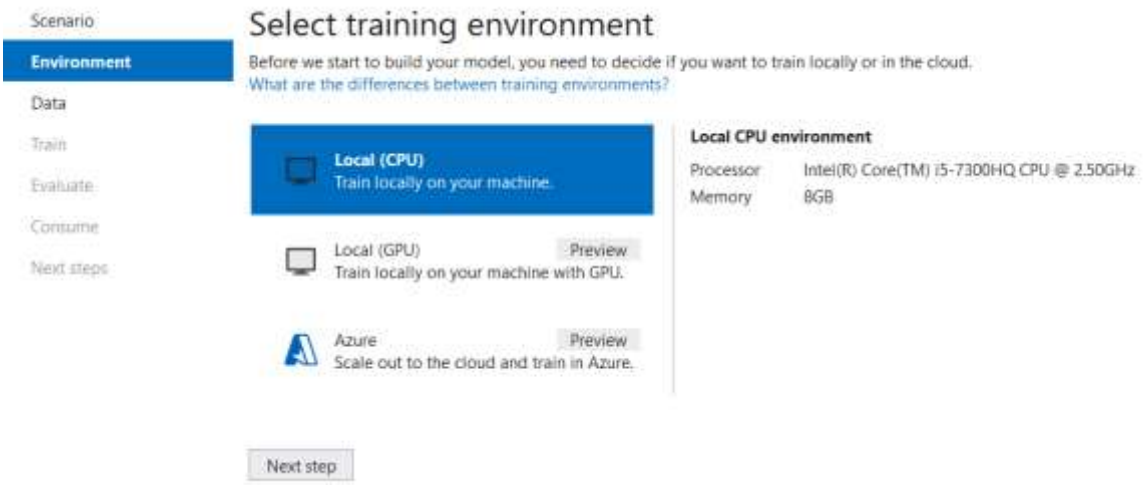
Slika 19: Veličina podataka spremnih za obradu (Izrada autora, 2021)

4.2. Implementacija

Prije samog početka implementacije, bitno je spomenuti kako je potrebno imati noviju verziju programa Visual Studio (2017. ili novije). Razlog tome je taj što ML.NET Model Builder nema podršku za ranije verzije Visual Studia. Za izradu prikazanog programskog rješenja u ovom završnom radu korišten je Visual Studio Enterprise 2022. Prvi korak implementacije je instalacija ekstenzije ML.Net Model Builder putem web stranice ili preko opcije „Ekstenzije“ u podatkovnoj traci. Nakon uspješne instalacije možemo kreirati novu windows form aplikaciju, u ovom slučaju s nazivom „Klasifikacija_zivotinja“. Nakon kreiranja projekta u Solution Exploreru kliknemo desnim klikom miša na naziv projekta, odaberemo „Add“ te naposljetku Machine Learning Model.

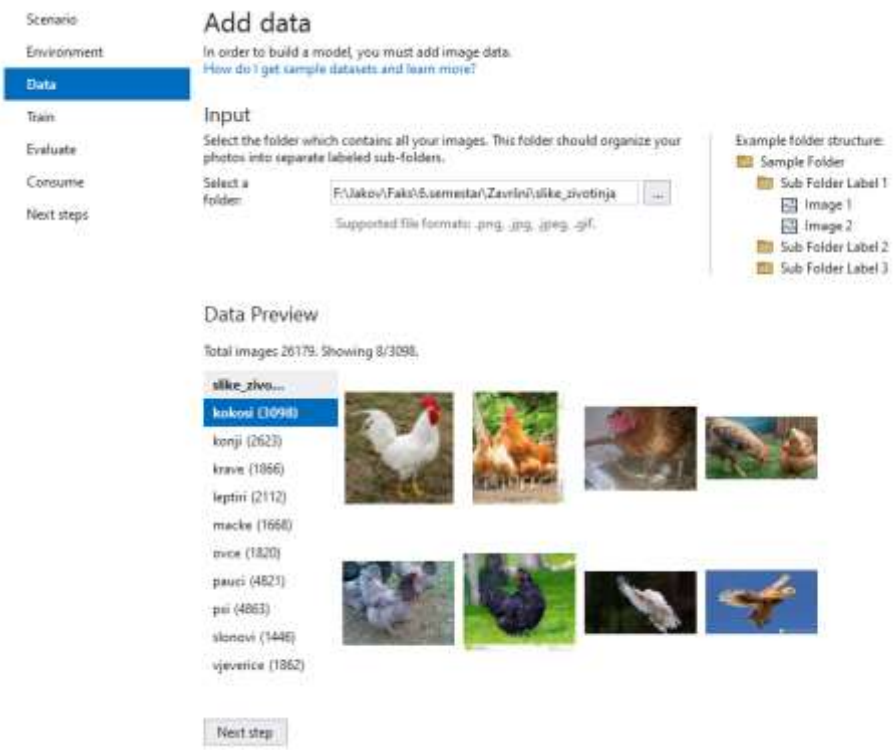
Slika 20: Početni zaslon Model Buildera (Izrada autora, 2022)

Otvara nam se početni zaslon u kojem možemo odabrati nekoliko scenarija. Ovo je vrlo korisna ekstenzija za izradu vlastitog modela strojnog učenja bez potrebe za velikim predznanjem strojnog učenja. Kako bi riješili problem završnog rada, odabire se scenarij: „Image classification“ odnosno klasifikacija slika u kategorije ili u ovom slučaju, predviđanje u koju skupina spada životinja sa slike. Budući da TensorFlow zna kako prepoznati uzorke sa slika, Model Builder koristi tu značajku za pretvaranje običnih sirovih slika u značajke za obuku.



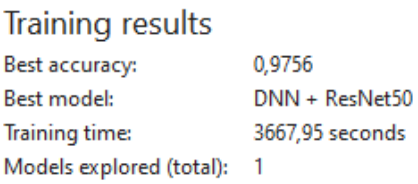
Slika 21: Odabir okruženja za treniranje (Izrada autora, 2022.)

Model Builder prilikom odabira scenarija klasifikacije slika omogućuje odabir 3 vrste okruženja. Lokalno (CPU), odnosno rad na procesoru, lokalno (GPU), gdje se rad izvršava na grafičkoj kartici i posljednja opcija je rad na Azure oblaku. Ako korisnik odabere Azure oblak, tada je korisniku potrebna Azure pretplata kako bi mogao izraditi radnu stanicu. Rad na grafičkoj kartici je omogućen korisnicima NVIDIA grafičkih kartica i NVIDIA.Developer računa. Završni rad izvršen je lokalno preko procesora.



Slika 22: Dodavanje podataka (Izrada autora, 2022.)

Posljednji korak prije samog testiranja je odabir testnih podataka. Potrebno je napomenuti kako slike životinja moraju biti u zasebnim datotekama kako ne bi došlo do miješanja označenih podataka. Nazivi datoteka u kojima se slike nalaze ujedno služe i kao nazivi kategorija u koje će se podaci klasificirati.



Slika 23: Rezultati testa (Izrada autora, 2022.)

Treniranje je završilo nakon 3667,95 sekundi, odnosno, 61.1 minutu te je ispisani sažetak istog. Najbolja točnost testiranja iznosi 0.9756 ili drugim riječima 97.56%. Kao najbolji model testiranja odabrani je DNN + ResNet50. DNN predstavlja duboku neuronsku mrežu dok je ResNet50 konvolucijska neuronska mreža dubine 50 slojeva. Premda smo u početku samog kreiranja testiranja odabrali model klasifikacije slika, istražen je samo jedan model.

Nakon testiranja dobivamo mogućnost generiranja kôda u Visual Studiu koji koristimo za izradu vlastitog programa za validaciju slika preko forme. Kôd je potrebno prilagoditi našim potrebama te nakon kratkih promjena dobivamo sljedeću klasu prikazanu na slici 24.

```
using MlClassifier;

namespace MLClassifier
{
    1 reference
    public static class Classifier
    {
        1 reference
        public static string Classify(string imagePath)
        {

            var imageBytes = File.ReadAllBytes(@$"{imagePath}");
            MLModel1.ModelInput sampleData = new MLModel1.ModelInput()
            {
                ImageSource = imageBytes,
            };
            var predictionResult = MLModel1.Predict(sampleData);
            return predictionResult.PredictedLabel;
        }

    }
}
```

Slika 24: Klasa Classifier (Izrada autora, 2022)

Važno je spomenuti da je klasa „*Classifier“* public, odnosno javna zato što nam je ona potrebna u drugom kôdu. Kada bi klasa ostala private, odnosno privatna, ona bi se mogla koristiti samo u kôdu u kojem je definirana, tj. ne bismo ju mogli pozvati i koristiti na drugim mjestima. Niz *Classify* stvara jednu instancu uzorka podataka iz prvog retka skupa podataka za unos modela, šalje tu instancu na predviđanje, što se može vidjeti na liniji 15. Instanca se prilikom klasifikacije učitava u unaprijed istrenirani model koji uspoređuje sliku s ostalim označenim podacima. Model traži određene specifikacije za određene životinje kako bi ih lakše i uspješnije prepoznao. Nakon predviđanja dobivamo povratnu informaciju o predviđenoj oznaci instance.

```
1   using Microsoft.ML;
2   using Microsoft.ML.Data;
3   namespace MlClassifier
4   {
        4 references | Jakov Gataric, 1 day ago | 1 author, 1 change
5       public partial class MLModel1
6       {
7           model input class
19
20          model output class
38
39          private static string MLNetModelPath = Path.GetFullPath("../../../../MLClassifier/MLModel1.zip");
40
41          public static readonly Lazy<PredictionEngine<ModelInput, ModelOutput>> PredictEngine =
42              new Lazy<PredictionEngine<ModelInput, ModelOutput>>(() => CreatePredictEngine(), true);
43
            1 reference | Jakov Gataric, 1 day ago | 1 author, 1 change
44          public static ModelOutput Predict(ModelInput input)
45          {
46              var predEngine = PredictEngine.Value;
47              return predEngine.Predict(input);
48          }
49
            1 reference | Jakov Gataric, 1 day ago | 1 author, 1 change
50          private static PredictionEngine<ModelInput, ModelOutput> CreatePredictEngine()
51          {
52              var mlContext = new MLContext();
53              ITransformer mlModel = mlContext.Model.Load(MLNetModelPath, out var _);
54              return mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);
55          }
56      }
57  }
```

Slika 25: Klasa MlModel1 (Izrada autora, 2022)

Na slici 25 možemo vidjeti auto generirani kôd od strane Model Buildera. Klasa „MLModel1“ služi za validaciju slike te je ona ključna za dobivanje ispravne oznake slike. Drugim riječima, kada unesemo sliku životinje, ova klasa prepozna kojoj skupini životinja ona pripada. Kako bi kôd radio prilikom preuzimanja s interneta potrebno je promijeniti putanju u kojoj se nalazi „MLModel1.zip“ datoteka, a to je vidljivo u liniji 39. Sada dolazimo do pitanja kako to sve spojiti da radi?

```
1   using MLClassifier;
2
3   namespace Ucitavanje_validacija_slike
4   {
        3 references
5       public partial class ValidacijaForm : Form
6       {
7           OpenFileDialog ucitajSliku = new OpenFileDialog();
8
            1 reference
9           public ValidacijaForm()
10          {
11              InitializeComponent();
12              prikazSlikePicBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.CenterImage;
13          }
14
            1 reference
15          public void ucitajButton_Click(object sender, EventArgs e)
16          {
17              ucitajSliku.Title = "Slika";
18              ucitajSliku.Filter = "slika(*.png;*.jpg;*.jpeg)|*.png;*.jpg;*.jpeg";
19              ucitajSliku.ShowDialog();
20              prikazSlikePicBox.Image = new Bitmap(ucitajSliku.FileName);
21          }
22
            1 reference
23          public void validirajButton_Click(object sender, EventArgs e)
24          {
25              var result = Classifier.Classify(ucitajSliku.FileName);
26
27              rezultatOutput.Text = "Rezultat klasifikacije je:\n "  + result.ToString();
28          }
29      }
30  }
```

Slika 26: Forma učitavanja i validacije slike (Izrada autora, 2022)

Ovo je programski kôd pomoću kojeg korisnik učitava sliku i dobiva ispis kojoj skupini životinja sa slike pripada. Javna klasa „*ValidacijaForm*“ u početku kreira novi *FileDialog* s nazivom „UcitajSliku“. Funkcija *OpenFileDialog* prikazuje standardni dijaloški okvir koji od korisnika traži da otvori neku datoteku ili u našem slučaju da odabere željenu sliku. Prilikom pritiska na gumb „ucitajButton“ postavljamo parametre za naslov na dijaloškom okviru te kakvu vrstu podataka tražimo. Zato što želimo validirati životinju sa slike, postavljamo da je tražena datoteka u obliku najčešćih ekstenzija za slike. Linija 19 označava otvaranje dijaloškog okvira, dok linija 20 prikazuje sliku koju je korisnik odabrao. Inicijalizacijom forme postavljamo parametre kako bi se učitana slika prikazala u centru što je vidljivo na liniji 12.

Korisnik ima opciju da pritisne tipku za validaciju te kada to učini šalje se naziv datoteke klasi „*Classifier*“ koja je već prethodno objašnjena. Nakon klasificiranja životinje, rezultat se prikazuje u obliku teksta na formi.

Slika 27: Početni izgled forme (Izrada autora, 2022)

Kao što je vidljivo na slici 27, forma je prazna te su vidljiva samo dva gumba, „Učitaj sliku“ i „Validacija“. Nakon pritiska na gumb „Učitaj sliku“ otvara se prozor u kojem odabiremo sliku koju želimo kasnije validirati. Ona se prikazuje na formi te pritiskom na gumb „Validacija“ ispisuje se kojoj grupi životinja ona pripada. Kao što je unaprijed navedeno, model treniranja prepoznaje skupine životinja prema nazivu datoteke u kojoj se one nalaze. Iz toga razloga predviđene oznake su u množini, iako je na slici jedna životinja.

Slika 28: Završni izgled forme (Izrada autora, 2022)

Naravno moguća je validacija više broja životinja bez izlaska iz forme te ponovnog ulaska. Iako je ovo gotov model, njega je moguće nadograditi dodavanjem drugih životinja, dodavanjem različitih pasmina kako bi on bio još točniji i precizniji. Ljepota, ali i prokletstvo modela strojnog učenja je ta što je on uvijek spreman za nadogradnju, novo učenje, poboljšanje performansi i slično. Cijelo programsko rješenje moguće je vidjeti na githubu: https://github.com/Wadell/Zavrsni_rad.

5. Zaključak

Strojno učenje je svuda oko nas i svakim danom se sve više i više razvija. Ono je s razlogom jedno od najbrže razvijenih grana informacijske tehnologije. U ovom radu upoznali smo se sa svom terminologijom umjetne inteligencije i strojnog učenja te na kraju uz primjer napravili programsko rješenje za otkrivanje koja je životinja na slici. U izradi rada pomogao je YouTube i Google za prikupljanje podataka. Važno je i spomenuti Microsoft ekstenziju Model Builder pomoću koje je bilo moguće izvršenje testiranja na bazi TensorFlow. Uveliko je pomogao auto generirani programski kôd kojeg je bilo potrebno prilagoditi za vlastite potrebe validacije slika. Najbolja točnost predviđanja životinje je 97% što je iznimno dobro kod klasifikacije. Kada bi se treniranje ponovilo nekoliko puta taj postotak bi došao bliže 100%.

Kroz ovaj rad naučili smo različite vrste strojnog učenja te njihovu primjenu, prednosti i nedostatke. Programsko rješenje je detaljno opisano, kako bi svaki čitatelj mogao razumjeti što se ovdje točno događa te možda i sam probati učiniti isto. Korist ovog sustava bi mogla biti značajna kada bi se unijele neke manje popularne životinje, neke ugrožene ili zaštićene kako ih ljudi ne bi ubijali iz straha i neznanja.

Smatram da u ovom projektu ima prostora za poboljšanje i nadogradnju, ali to i jest čar umjetne inteligencije. Ona uvijek trenira, uči nove stvari i spremna je na svaki problem na koji naiđe. Iako je umjetna inteligencija već prisutna u našem svakodnevnom životu te nam stvarno mnogo pomaže u različitim sektorima života i rada, tempom kojim se razvija, mislim da kroz nekoliko godina, život bez nje neće biti moguć.

Popis literature

.NET Microsoft. (2022). *ML.NET Model Builder*. Dohvaćeno iz https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet/model-builder

Alpaydin E. (2021). *Strojno učenje: nova umjetna inteligencija.* Mate d.o.o.

Altexsoft. (2022). *Traffic Prediction: How Machine Learning Helps Forecast Congestions and Plan Optimal Routes*. Dohvaćeno iz https://www.altexsoft.com/blog/traffic-prediction/

Asiri S. (2018). *Machine Learning Classifiers*. Dohvaćeno iz https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

Bewtra A. (2022). *The Ultimate Guide to Semi-Supervised Learning*. Dohvaćeno iz https://www.v7labs.com/blog/semi-supervised-learning-guide

Chollet F. (2017). *Deep Learning with Python.* Manning Publication.

Datrics. (2022). *AI in Stock Trading: What you need to know in 2020*. Dohvaćeno iz https://datrics.ai/ai-in-stock-trading-what-you-need-to-know-in-2020

Defined.AI. (2022). *A 2022 Speech AI Guide: Key Application Methods and Technologies*. Dohvaćeno iz https://www.defined.ai/blog/what-is-speech-recognition/?WPACFallback=1&WPACRandom=1659672279421

Duggal N. (2022). *Top 10 Machine Learning Applications and Examples in 2022*. Dohvaćeno iz https://www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-applications

Florian. (2022). *Fraud Detection with Machine Learning & AI*. Dohvaćeno iz https://seon.io/resources/fraud-detection-with-machine-learning/

Gandhi R. (2018). *Introduction to Machine Learning Algorithms: Linear Regression*. Dohvaćeno iz https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a

GeeksforGeeks. (2022). *Advantages and Disadvantages of TensorFlow*. Dohvaćeno iz https://www.geeksforgeeks.org/advantages-and-disadvantages-of-tensorflow/

GeeksforGeeks. (2022). *Clustering in Machine Learning*. Dohvaćeno iz https://www.geeksforgeeks.org/clustering-in-machine-learning/

Heidenreich H. (2018). *What are the types of machine learning?* Dohvaćeno iz https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f

IBM. (2020). *Unsupervised Learning*. Dohvaćeno iz https://www.ibm.com/cloud/learn/unsupervised-learning

IBM. (2022). *Neural network*. Dohvaćeno iz https://www.ibm.com/cloud/learn/neural-networks

javaTpoint. (2021). *Applications of Machine learning*. Dohvaćeno iz https://www.javatpoint.com/applications-of-machine-learning

javaTpoint. (2021). *Types of Machine learning*. Dohvaćeno iz https://www.javatpoint.com/types-of-machine-learning

Kelleher J.D., Tierney B. (2021). *Znanost o podacima.* Mate d.o.o.

Korstanje J. (2021). *The eclat algorithm*. Dohvaćeno iz https://towardsdatascience.com/the-eclat-algorithm-8ae3276d2d17

Korstanje J. (2021). *THE FP Growth algorithm*. Dohvaćeno iz https://towardsdatascience.com/the-fp-growth-algorithm-1ffa20e839b8

Lutkevich B. (2019). *Self driving car (autonomous car or driverless car*. Dohvaćeno iz https://www.techtarget.com/searchenterpriseai/definition/driverless-car#:~:text=A%20self%2Ddriving%20car%20(sometimes,destinations%20without%20a%20human%20operator

Microsoft .NET. (2022). *ML.NET*. Dohvaćeno iz https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet

Microsoft Azure. (2022). *Artificial intelligence(AI) vs Machine learning(ML)*. Dohvaćeno iz https://azure.microsoft.com/en-us/solutions/ai/artificial-intelligence-vs-machine-learning/#introduction

Nielsen M. (2021). *Neural Networks and Deep Learning*. Dohvaćeno iz https://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf

Nikolaiev, D. (2022). *Unsupervised Learning algorithms cheat sheet*. Dohvaćeno iz https://towardsdatascience.com/unsupervised-learning-algorithms-cheat-sheet-d391a39de44a

Nilesh B. (2022). *Dimensionality Reduction for Machine Learning*. Dohvaćeno iz https://neptune.ai/blog/dimensionality-reduction

PCChip. (2021). *Razlike između strojnog učenja i dubokog učenja*. Dohvaćeno iz https://pcchip.hr/ostalo/tech/razlike-izmedu-strojnog-ucenja-ai-i-dubokog-ucenja/

Pedamkar P. (2022). *Introduction to Tensorflow*. Dohvaćeno iz https://www.educba.com/introduction-to-tensorflow

Potentia. (2021). *What Is Machine Learning: Definition, Types, Applications and Examples*. Dohvaćeno iz https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/

Raschka S., M. V. (2017.). *Python Machine Learning: Machine learning and Deep Learning with Python, scikit-learn and TensorFlow.* Packt Publishing.

Sarangam A. (2022). *Different Types of Regression Analysis - A Basic Guide*. Dohvaćeno iz https://www.jigsawacademy.com/blogs/data-science/types-of-regression-analysis/

Schiavini, R. (2019). *How does AI work with product recommendation systems?* Dohvaćeno iz https://www.smarthint.co/en/ai-product-recommendation-engine/

Schmitt M. (2022). *Artificial Intelligence in Medicine*. Dohvaćeno iz https://www.datarevenue.com/en-blog/artificial-intelligence-in-medicine

Shalev-Shwartz S., B.-D. S. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press.

Smola A., V. S. (2008). *Introduction to Machine Learning.* Cambridge University Press.

TensorFlow. (2022). *Introduction to graphs and tf.function*. Dohvaćeno iz https://www.tensorflow.org/guide/intro_to_graphs

TensorFlow. (2022). *TensorFlow*. Dohvaćeno iz https://www.tensorflow.org/

TensorFlow. (2022). *TensorFlow Cloud*. Dohvaćeno iz https://www.tensorflow.org/cloud/tutorials/overview

TensorFlow. (n.d.). *Deploy machine learning models on mobile and edge devices*. Dohvaćeno iz https://www.tensorflow.org/lite

Veel M. (2022). *Image Recognition: The Basics and Use Cases (2022 Guide)*. Dohvaćeno iz https://viso.ai/computer-vision/image-recognition/

Vijesti Europski parlament. (2021). *Što je umjetna inteligencija i kako se upotrebljava?* Dohvaćeno iz https://www.europarl.europa.eu/news/hr/headlines/society/20200827STO85804/sto-je-umjetna-inteligencija-i-kako-se-upotrebljava

Yegulalp S. (2022). *What is TensorFlow? The machine learning library explained*. Dohvaćeno iz https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

Popis slika

Popis tablica

Prilozi radu

Programsko rješenje: https://github.com/Wadell/Zavrsni_rad

Slike životinja: https://mega.nz/file/Zl5n0BrR#sWxN3kfbu4_LV-6C32nO1znFK3phlNztvG-UdbkJsF0