Prepoznavanje bolesti palmine pipe (Rhynchophorus ferrugineus)

Gatarić, Filip

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:268122

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Filip Gatarić

PREPOZNAVANJE BOLESTI PALMINE PIPE (RHYNCHOPHORUS FERRUGINEUS)

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Gatarić

Matični broj: 42676/13–R

Studij: Informacijsko i programsko inženjerstvo

PREPOZNAVANJE BOLESTI PALMINE PIPE (RHYNCHOPHORUS FERRUGINEUS)

DIPLOMSKI RAD

Mentor/Mentorica:

Prof. dr. sc. Kliček Božidar

Varaždin, rujan 2020.

Filip Gatarić

Izjava o izvornosti

Izjavljujem da je moj diplomski 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/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

---

Sažetak

Glavni cilj ovog rada je upoznati se s osnovama strojnog učenja te opisati i prikazati tehnike i metode strojnog prepoznavanja uzoraka kojima je moguće prepoznati prisutnost crvene palmine pipe na određenoj palmi. Crvena palmina pipa je nametnik koji je u Hrvatskoj prvi puta opažen 2011. godine i od onda uništava palme po cijeloj našoj jadranskoj obali. Postoje različiti tretmani prevencije palmine pipe, ali je najveći problem što se početna faza infestacije može uočiti na mladim listovima palme, unutar krune palme, što je problem za uočiti s obzirom da je palmina krošnja visoko iznad tla. Glavna misao vodilja ovog rada je da bi se prelaskom i snimanjem palmine krune iz zraka, pomoću bespilotne letjelice, omogućilo rano otkrivanje prisutnosti nametnika i samim time spašavanje palme od umiranja. Snimka koju bespilotna letjelica snima se analizira pomoću kreiranog modela i vraća rezultat analize, da li je palma zaražena i koji je postotak točnosti da je palma zaražena. Model koji će se koristiti u ovom radu je binarni klasifikacijski model o čijem modeliranju i treniranju će biti više riječi u narednim poglavljima. Aplikacija i binarni klasifikacijski modeli strojnog učenja za prepoznavanje bolesti palmine pipe biti će kreirani pomoću Python programskog jezika.

Ključne riječi: strojno učenje, prepoznavanje uzoraka, klasifikacijski model, binarni klasifikacijski model, umjetna inteligencija, veliki podaci, bespilotna letjelica, palma, crvena palmina pipa

Sadržaj

1. Uvod

Kada se ljudima kaže da zamisle savršeni odmor, većina će ljudi nacrtati istu mentalnu sliku. Zamisliti će kako leže ispod palme na pješčanoj plaži pokraj modrog mora i s koktelom u ruci. Što ako te palme nema jer su je ubili nametnici zvani crvena palmina pipa? Ovaj će se rad fokusirati na eksperimentalnu metodu prepoznavanja bolesti palmine pipe snimanjem krune palme iz zraka bespilotnom letjelicom. Prisutnost nametnika će se prepoznavati treniranim modelom strojnog učenja.

Kroz rad će se upoznati sa softverskim rješenjima i hardverskim komponentama s kojima se kreirao sustav za prepoznavanje bolesti palmine pipe. Predstavit će se i nametnik čiju prisutnost u palmi pokušavamo prepoznati. Opisati će se teorijski predlošci strojnog prepoznavanja uzoraka, te eksperimentalno potvrditi u aplikaciji. Proces implementacije aplikacije biti će popraćen detaljnim komentarima i vizualizacijom modela i rezultata treniranja modela.

2. Metode i tehnike rada

Pošto je cijeli projekt rađen u Windows 10 operacijskom sustavu, korištena je aplikacija Anaconda Navigator i Visual Studio Code za rad sa Python-om. Anaconda omogućuje pokretanje aplikacija i jednostavno upravljanje conda paketima, okruženjima i kanalima bez korištenja naredbi iz terminala, a dostupna je za Windows, macOS i Linux operacijske sustave. Za implementaciju programa za rješavanje problema otkrivanja bolesti palmine pipe odabran je programski jezik Python zbog velike zbirke svojih specijaliziranih biblioteka za razne potrebe, uključujući biblioteke za strojno učenje i računalni vid.

2.1. Keras

Za izradu modela, korištena je biblioteka Keras. Keras je biblioteka neuronskih mreža otvorenog koda (eng. *open source*) napisana na Python-u. Može se pokretati preko TensorFlow, Cognitive Toolkit, R, Theano ili PlaidML. Dizajniran je kako bi omogućio brzo eksperimentiranje s dubokim neuronskim mrežama, fokusira se na dobro korisničko iskustvo (eng. *user-friendly*), modularnost i proširivost. Keras omogućuje izgradnju raznih vrsta modela i pokretanje na CPU i GPU jedinicama. Ovaj rad je napravljen na Keras verziji koja je namijenjena za pokretanje na Nvidia grafičkim karticama. [1]

Uz Keras, za realizaciju aplikacije, korištene su i biblioteke Numpy za matematičke operacije, OpenCV za računalni vid, os za funkcionalnosti koje su ovisne o operacijskom sustavu i Matplotlib za vizualizaciju podataka.

2.2. NumPy

NumPy je biblioteka za programski jezik Python koja omogućuje podršku velikim, višedimenzionalnim nizovima i matricama, zajedno s velikom zbirkom matematičkih funkcija visoke razine za rad na tim nizovima, a isto tako je i neizostavna biblioteka kada se radi sa bilo kakvom obradom podataka. Omogućuje kreiranje polja, osnovnih matematičkih operacija i univerzalnih funkcija, linearnu algebru, a što je nama jako interesantno podržava tenzore i uključuje se u OpenCV. [2]

2.3. OpenCV

OpenCV je biblioteka programskih funkcija uglavnom usmjerenih na računalni vid u stvarnom vremenu. Biblioteka je više-platformska i besplatna za upotrebu.OpenCV je napisan u C ++ jeziku i primarno sučelje mu je također na C ++ jeziku. Iako je primarno bio namijenjen za C++, postoje verzije i za Python, Java, MATLAB, a od verzije 3.4 i za JavaScript web platforme. OpenCV-Python koristi NumPy, visoko optimiziranu biblioteku za numeričke operacije sa sintaksom u stilu MATLAB-a. Sve strukture OpenCV polja pretvaraju se u i iz NumPy nizova.

To također olakšava integraciju s drugim bibliotekama koje koriste NumPy, kao što su SciPy i Matplotlib. [3]

2.4. Matplotlib

Matplotlib je biblioteka "crtanja podataka" za programski jezik Python i njegovo numeričko matematičko proširenje NumPy. Pruža objektno orijentirani API za ugrađivanje ploha u aplikacije pomoću GUI-a opće namjene kao Tkinter, wxPython, Qt ili GTK +. Tu je i proceduralno "pylab" sučelje temeljeno na stroju stanja (poput OpenGL-a), dizajnirano da sliči na ono MATLAB-a. U projektu se najviše koristila pyplot zbirka funkcija.

Matplotlib.pyplot je zbirka funkcija naredbenog stila. Svaka funkcija pyplot-a vrši neke promjene na slici: npr. stvara lik, stvara područje crtanja na slici, iscrtava neke crte u području crtanja, ukrašava platno oznakama i tako dalje. U matplotlib.pyplot u pozivima funkcija su sačuvana različita stanja, tako da prati stvari poput trenutne slike i područja crtanja, a funkcije crtanja usmjerene su na trenutne osi (treba imati na umu da su "osi" ovdje i na većini mjesta u dokumentacija odnose na dio osi slike, a ne na strogi matematički pojam za više osi). [4]

2.5. python os

Ovaj Python modul omogućuje korištenje funkcionalnosti koje su ovisne o operativnom sustavu. Ako samo želimo pročitati ili napisati datoteku, to možemo učiniti naredbom open (), ako želimo manipulirati putanjama, to možemo raditi os.path modulom, a ako želite pročitati sve retke u svim datotekama na naredbenom retku, pogledajte modul za unos datoteke .Os nam također omogućuje i kreiranje/ brisanje datoteka i direktorija i mnoge druge funkcionalnosti. [5]

2.6. DJI Phantom 3 Standard

Za prikupljanje fotografija iz zraka, korištena je bespilotna letjelica DJI Phantom 3 Standard. Bespilotna letjelica (eng. *drone*) je letjelica koja nema posadu, ali koja se može nadzirati na daljinu ili može samostalno letjeti prema programiranom planu. S obzirom da je Phantom 3 Standard bespilotna letjelica za početnike, iznenadila me je kvaliteta videozapisa koje snima sa svojom kamerom.

Kamera snima videozapise kvalitete 2.7k: 2701x1520p i automatski stabilizira kameru, a samim time i snimku. Rezultat su videozapisi visoke kvalitete što omogućuje preuzimanje fotografija visoke razine čistoće i vidljivosti detalja iz navedenih videozapisa. Snimanje palminih krošnji iz zraka, a zatim preuzimanje fotografija iz videa je uvelike olakšalo proces prikupljanja podataka. [6]

Slika 1: DJI Phatnom 3 standard bespilotna letjelica (Izvor: https://www.dji.com/hr/phantom-3-standard, 2020)

2.7. NVIDIA Quatro P1000

Nvidia Quadro P1000 je mobilna grafička kartica radne stanice početne razine za prijenosna računala. Quadro GPU-ovi nude certificirane upravljačke programe koji su optimizirani za stabilnost i performanse u profesionalnim aplikacijama (CAD, DCC, medicinski, prospekcijski i vizualizacijski programi). Izvedbe u tim područjima su stoga puno bolje u usporedbi s odgovarajućim potrošačkim GPU-ima. Ispostavilo se da je procesna snaga ove grafičke kartice bila dovoljna za treniranje modela ali se treniranje modela odvijalo jako sporo, za pojedine modele i preko 5 sati. Kako bi provjerili da se za procesiranje uistinu koristi ovaj GPU, izvodimo naredbu

```
import tensorflow as tf;
print(tf.reduce_sum(tf.random.normal([1000, 1000])))
```

i prema odgovoru kojeg dobijemo, možemo vidjeti da se stvarno koristi Quadro P1000 kao fizički GPU.

```
Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 2999 MB
     memory) -> physical GPU (device: 0, name: Quadro P1000, pci bus id:
    0000:01:00.0, compute capability: 6.1)
tf.Tensor(520.50916, shape=(), dtype=float32)
```

3. Crvena palmina pipa

Crvena palmina pipa (lat. *Rhynchophorus ferrugineus*) je jedan od najvećih štetnika palmi na području Europe. Ovaj štetnik južno-azijskog podrijetla počeo se širiti prema zapadu sredinom osamdesetih godina dvadesetog stoljeća, a na Mediteranu je prvi puta opažen 1992. godine u Egiptu. Glavni uzrok ubrzanog širenja ovih štetnika je prvenstveno međunarodna trgovina zaraženim sadnicama. Pojavom crvene palmine pipe u Europi došlo je do masovnog propadanja palmi, ponajviše zbog nepoznavanja biologije ovih štetnika i njihovog suzbijanja. Hraneći se, ličinke crvene palmine pipe djeluju devastirajući, bušeći hodnike u deblu palme i jedući mlade izdanke palminih listova. Njihova ishrana rezultira postupnom propadanju palme, koje dugo vremena ostaje prikriveno i samim time neuočljivo što na kraju dovodi do potpunog propadanja i izumiranja palme.

Najčešće žrtve napada su palme iz porodice Arecaceae koje dosta često susrećemo na našem području, a potencijalni domaćini ovih štetnika su također i kanarska datulja (lat. *Phoenix canariensis*), visoka žumara (lat. *Trachycarpus fortunei*), končasta palma (lat. *Washingtonia filifera*) i niska žumara (lat. *Chamaerops humilis*). Crvena palmina pipa osobito velike štete izaziva na kanarskim datuljama.

Crvena palmina pipa je kukac poprilično velikih dimenzija. Odrasla jedinka crvene palmine pipe ima izduženo ovalno tijelo, crvenkasto smeđe do crne boje, prosječne duljine 2-4 cm. Ima dugačko rilo. Ličinka je kruškolike forme, bez nogu, blijedo žućkaste boje, prosječne duljine 3.5-4.5 cm. Glava joj je crvenkasto-smeđe do sjajno smeđe crne boje. [7]

3.1. Životni ciklus

Crvena palmina pipa ima 2-3 generacije godišnje. U istom stablu palme može biti nazočno više generacija u različitim razvojnim stadijima. Vrhunac leta crvene palmine pipe odvija se u listopadu i početkom studenog. U pravilu, odrasle pipe prisutne u jednom stablu neće se seliti na drugo sve dok imaju dovoljno hrane. Ove kukce u prvom redu privlače odumirući ili oštećeni dijelovi palmi, ali je moguće da i neoštećene palme budu napadnute.

Nakon oplodnje odrasla ženka može položiti između 300 i 500 jajašaca. Jaja leže u rupe koje su produkt nametnikove potrage za hranom ili pak koriste pukotine ili rane na nedavno posječenim palmama . Nakon polaganja, ženka štiti i osigurava jajašca sekretom koji se brzo stvrdne oko jajašaca. U prosjeku ženke proizvedu 210 jajašaca po leglu, od kojih se većina izleže u razdoblju od 3 dana. Jaja su bijela, cilindrična, sjajna, ovalnoga oblika i dimenzija su 1-2.5 mm.

Ličinke novorođenčadi su žuto-bijele, segmentirane, bez nogu i imaju hitinsku kapsulu glave koja je tamnije smeđa od ostatka tijela. Imaju moćne vodoravne stožaste čeljusti kojima se udubljuju od vrhova lišća do tjemena, gdje se proždrljivo hrane. Po završetku razvoja ličinki, ličinka će ponekad izroniti iz debla stabla i izgraditi kukuljicu vlakana izvučenih iz debla palme. Ličinka će tada proći metamorfozu u odraslu osobu. Ličinka će također tkati kukuljicu na dnu palminih listova ili u središtu baze biljke.

Nakon što ličinka izađe iz jaja, zavlači se u peteljku lista bušenjem hodnika i započinje se hraniti u unutrašnjosti palme. Dok se hrane, ličinke stvaraju masu koja ispunjava hodnike nastale hranjenjem, a sastoji se od sažvakanih biljnih vlakana i biljnog soka.

Odrasle jedinke su izvrsni letači i sposobni je putovati na velike udaljenosti. Iako više vole napadati palme koje su već zaražene ili oslabljene drugim čimbenicima, kolonizirati će i zdrave palme. [8] Životni ciklus po fazama crvene palmine pipe može se vidjeti na slici 2, dok se na slici 3 može dobiti percepcija o veličini nametnika i njegovih čahura.

Slika 2: Životni ciklus po fazama crvene palmine pipe (Izrada autora prema [8], 2020)

Slika 3: Učahurene ličinke i odrasla jedinka palmine pipe (Izvor: Damir Ivačić, 2019)

3.2. Šteta i simptomi napada

Simptomi napada će se prikazati u vizualno, putem slika, pošto je primarni cilj izrade ovog rada prepoznavati prisutnost palmine pipe na palmi bitno je kako simptomi napada izgledaju, a ne kako i zašto do njih dolazi. Simptomi napada crvene palmine pipe su sljedeći: [8]

- Piljevina na kruni ili deblu palme (Slika 4)

Slika 4: Piljevina na kruni ili deblu palme (Izvor: Damir Ivačić, 2019)

- Prisutnost perforiranih ili izgriženih listova (Slika 5)

Slika 5: Perforirani i izgriženi listovi (Izvor: Damir Ivačić, 2019)

- Prisutnost dugačkih hodnika unutar palminih listova ili debla (Slika 6)

Slika 6: Prisutnost dugačkih hodnika unutar palminih listova i debla (Izvor: Damir Ivačić, 2019)

- Deformirano i abnormalno savijanje debla palme (Slika 7)

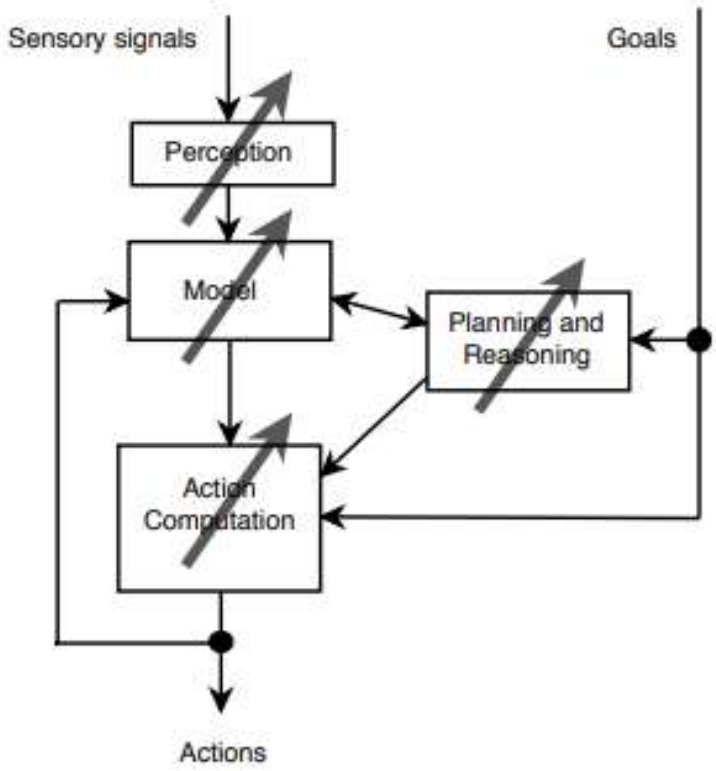
Slika 7: Deformirano deblo palme (Izvor: Damir Ivačić, 2019)

- Sušenje palme (Slika 8)

Slika 8: Suhe palme i palme pred umiranje (Izvor: Damir Ivačić, 2019)

4. Strojno učenje

Tijekom posljednja dva desetljeća strojno učenje (eng. *machine learning*) postalo je jedno od najbrže rastućih grana informatike i jedno od glavnih oslonaca informacijske tehnologije, a samim time, iako obično sakriveni, dio našeg života. Sa sve većom količinom podataka koja se svakodnevno sakuplja, gomila i postaje dostupna, jako je dobar razlog za predviđanje i vjerovanje da će pametna analiza podataka uz pomoć strojnog učenja postati još traženiji i nužniji sastojak tehnološkoga napretka informatičke tehnologije.[9]



Slika 9: Arhitektura sustava umjetne inteligencije (Izrada autora prema [10], 2005)

Pojam „strojno učenje“ odnosi se na automatizirano otkrivanje smislenih obrazaca u podacima. U posljednjih par desetljeća postao je uobičajeni alat u gotovo svakom zadatku koji zahtijeva izdvajanje podataka iz velikih skupova podataka (eng. *big data*). Mi smo okruženi tehnologijom temeljenom na strojnom učenju i prečesto ju uzimamo zdravo za gotovo ili pak niti nismo svjesni u kolikoj su količini te tehnologije uključene u naš svakodnevni život. Neki od primjera primjene strojnog učenja iz svakodnevnog života su :[11]

- tražilice koje uče kako bi nam dale najbolje rezultate (uz postavljanje profitabilnih oglasa)
- softver protiv neželjene pošte uči filtrirati naše e-poruke
- transakcije kreditnim karticama su osigurane softverom koji uči kako otkriti prijevare
- digitalni fotoaparati uče otkrivati lica

- inteligentne aplikacije za osobnu pomoć na pametnim telefonima nauče prepoznati glasovne naredbe
- filteri za fotografije i videozapise na društvenim mrežama
- automobili opremljeni sustavima za sprečavanje nesreća
- znanstvene primjene poput bio informatike, medicine i astronomije

Iz priloženih primjera može se uočiti da su aplikacije strojnog učenja jako široke i primjenjive u bilo kojem području našeg privatnog i poslovnog života. Jedna zajednička značajka svih ovih aplikacija je ta, za razliku od više tradicionalnih uporaba računala, zbog složenosti obrazaca koje treba uočiti, ljudski programer ne može pružiti izričitu i detaljnu specifikaciju načina definiranja i izvršavanja takvih zadataka. Uzimanje primjera iz inteligentnih bića, mnoge naše vještine stječu se ili usavršavaju kroz učenje iz vlastitog iskustva (umjesto da slijedimo izričite upute koje su nam dane). Alati temeljeni na strojnom učenju bave se obdarivanjem programa sposobnošću „učenja“ prilagođavanja novim situacijama temeljem svojih dosadašnjih „iskustava“. [11]

Ulazni podaci u algoritam učenja su podaci za trening koji predstavljaju iskustvo, a rezultat je neka stručnost ili znanje koja obično ima oblik drugog računalnog programa koji može izvršiti određeni zadatak. Tražeći formalno-matematičko razumijevanje ovog koncepta, morati ćemo biti eksplicitniji o tome što podrazumijevamo pod svakom od uključenih pojmova: Koji su podaci za trening kojima će naši programi pristupiti? Kako može proces učenja biti automatiziran? Kako možemo procijeniti uspjeh takvih procesa? [11]

4.1. Što je učenje?

Mi želimo programirati računala tako da oni mogu "učiti" iz ulaznih podataka koji su im dostupni. Ali kako? Što je zapravo učenje? Prema Richard Gross-u učenje je proces stjecanja novog razumijevanja, znanja, ponašanja, vještina, vrijednosti, stavova i preferencija temeljem ponavljajućih iskustava. Grubo govoreći, učenje je postupak pretvaranja iskustva u stručnost ili znanje. [12]

Kako bismo mogli razumjeti na koji način računalo uči, krenimo od razmatranja jednog od primjera iz učenja životinja na prirodan način. Neke od najvažnijih pitanja u području strojnog učenja već se pojavljuju u tim kontekstima, koji su nam svima poznati i lako shvatljivi.

Štakori koji uče izbjegavati otrovne mamce: Kada se štakori susretnu s prehrambenim proizvodima novog izgleda ili mirisa, prvo će pojesti vrlo male količine, dok će svako sljedeće hranjenje ovisit o okusu hrane i njezinim fiziološkim utjecajima na tijelo životinje. Ako hrana rezultira lošim učinkom, ta nova hrana često će biti povezana s bolešću, a nakon toga je štakori više neće ni jesti. Jasno je da je ovdje u igri određeni mehanizam učenja - životinja je s koristila svoja prethodna iskustva s hranom za stjecanje stručnosti i znanja u otkrivanju sigurnosti konzumacije nove hrane. Ako je prošlo iskustvo s hranom bilo označeno kao negativno, životinja

predviđa da će ista ta hrana imati negativan učinak i kada se s njom susretne u budućnosti. [11]

Inspirirani prethodnim primjerom uspješnog učenja, pokažimo tipični zadatak strojnog učenja. Pretpostavimo da bismo željeli programirati sustav koji uči kako filtrirati neželjenu e-poštu (eng. *spam*). Prvo riješenje koje bi nam palo na pamet, koje je ujedno i naivno rješenje, bilo bi naizgled slično na način na koji štakori uče kako izbjeći otrovne mamce. Stroj bi jednostavno memorirao sve prethodne e-mailove koji su označeni kao neželjena e-pošta od strane ljudskog korisnika. Kad stigne nova e-pošta, uređaj će je potražiti u bazi podataka prethodno neželjenih e-mailova. Ako se podudara s jednim od njih, bit će uništen. Inače, premjestit će se u mapu ulazne pošte korisnika. Iako je prethodni pristup „učenja memoriranjem“ ponekad koristan, nedostaje mu važan aspekt sustava učenja - sposobnost označavanja neotvorenih e-mail poruka. Uspješan učenik trebao bi moći napredovati od pojedinačnih primjera do šire generalizacije. Ovakvo procesiranje podataka i zaključivanje temeljeno na njima se naziva i induktivno zaključivanje. [11]

Za postizanje generalizacije u zadatku filtriranja neželjene pošte, učenik može proći kroz prethodno otvorene e-mailove i izvoditi skup riječi čija pojava u e-mailu indicira na neželjenu poštu. Zatim, kada stigne novi e-mail, sustav može provjeriti pojavljuje li se u primljenom e-mailu jedna od sumnjivih riječi iz seta i u skladu s tim predviđa njezinu oznaku(željena ili neželjena pošta). Takav sustav bi potencijalno mogao pravilno predvidjeti oznaku neotvorene e-mail pošte. [11]

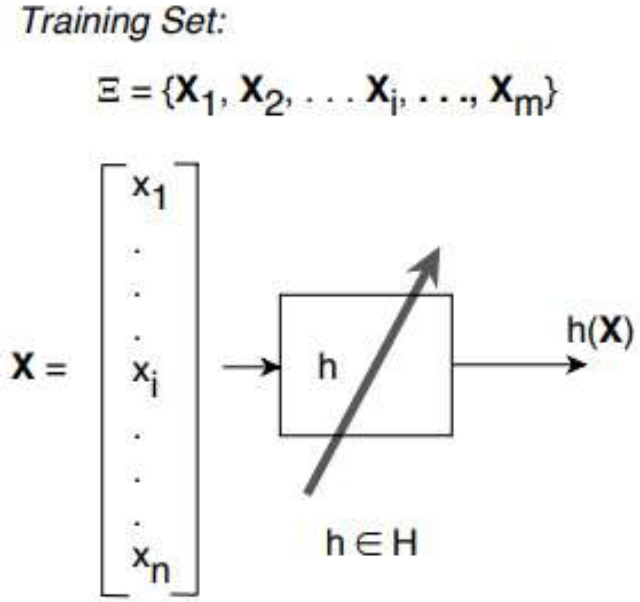
Ispada da su uključivanje predznanja i pristranost u procesu učenja neizbježni faktori za uspješne algoritme učenja. Razvojni alati za izražavanje stručne domene, prevođenje iste u pristranost učenja i kvantificiranje učinaka takve pristranosti na uspjeh učenja središnja je tema teorije strojnog učenja. Grubo rečeno, što je jače prethodno znanje ili prethodne pretpostavke, s kojim započinje proces učenja, lakše će biti učiti iz budućih primjera. [11]

4.2. Zašto nam je potrebno strojno učenje?

Mi kao ljudi obavljamo veliku količinu rutinskih poslova, ali naša introspekcija s time kako ih mi obavljamo nije dovoljno razrađena da se izvadi dobro definiran program. Neki od primjera takvih zadataka su vožnja, govor, prepoznavanje i razumijevanje slike. U svim tim zadacima, vrhunski programi koji se koriste tehnikama strojnog učenja postižu sasvim zadovoljavajuće rezultate, nakon što su izloženi velikom broju kvalitetnih primjera treninga.

Neki se zadaci ne mogu dobro definirati, osim primjerom; to jest, mogli bismo biti sposobni odrediti ulazno / izlazne parove, ali ne i sažeti odnos između ulaznih podataka i željenih izlaza. Željeli bismo da strojevi mogu prilagoditi svoju unutarnju strukturu za stvaranje točnih rezultata za velik broj ulaznih podataka i na taj način prikladno ograničiti svoju ulazno / izlaznu funkciju (Slika 10) za odnos implicitan s primjerima.

Moguće je da se među velikim hrpama podataka skrivaju važni odnosi i korelacije koje ručna ljudska analiza nikada ne bi uočila. Često se mogu koristiti metode strojnog učenja za izdvajanje tih odnosa poznatije pod nazivom rudarenje podacima (eng. *data mining*) .



Slika 10: Ulazno - izlazna funkcija (Izrada autora prema [10], 2005)

Ljudski dizajneri često proizvode strojeve koji ne rade optimalno u željenim okruženjima u kojima se koriste. Zapravo, određene karakteristike radne okoline možda neće nti biti potpuno poznate za vrijeme dizajniranja. Metode strojnog učenja mogu se koristiti za poboljšanje dizajna postojećih strojeva.

Količina dostupnog znanja o određenim zadacima može biti prevelika za eksplicitno kodiranje od strane ljudi. Strojevi koji uče to znanje će postupno moći uhvatiti više podataka nego što bi to ljudi htjeli zapisati.

Okruženja se vremenom mijenjaju. Strojevi koji se mogu prilagoditi promjeni okoline smanjili bi potrebu za stalnim redizajnom.

Ljudi neprestano otkrivaju nova znanja o zadacima. Promjene rječnika. U sustavu. Postoji stalni tok novih događaja u svijetu. Kontinuirano redizajniranje sustava umjetne inteligencije u skladu s novim znanjima je poprilično nepraktično, ali metode strojnog učenja možda mogu pratiti i biti u toku s većinom novih saznanja.

Jedna od ograničavajućih značajka programiranih alata je njihova rigidnost - jednom kada je program zapisan i instaliran, ostaje nepromijenjen. Međutim, mnogi se zadaci mijenjaju tijekom vremena ili od jednog korisnika do drugog. Alati strojnog učenja - programi čije se ponašanje prilagođava njihovim ulaznim podacima - nudi rješenje za takve probleme. Oni su zapravo po prirodi prilagodljivi na promjene u okolini s kojom komuniciraju. Tipični promjeri uspješne primjene strojnog učenja na takve probleme uključuje programe koji su u mogućnosti dekodirati tekst pisan rukom, gdje se fiksni program može prilagoditi varijacijama između rukopisa različitih korisnika, programi za otkrivanje neželjene pošte, automatsko prilagođavanje promjenama u prirodi neželjene e-pošte i programi za prepoznavanje govora. [10]

4.3. Tipovi učenja

Učenje je vrlo široka domena. Sukladno tome, područje strojnog učenja se razgranalo u nekoliko podpolja koja se bave različitim vrstama i metodama zadataka učenja. U nastavku slijedi gruba taksonomija paradigmi učenja. Budući da učenje uključuje interakciju između učenika i okoline, zadaće učenja mogu se podijeliti prema prirodi te interakcije. Prva razlika koju treba primijetiti je razlika između učenja pod nadzorom (eng. *supervised learning*) i bez nadzora (eng. *unsupervised learning*).

4.3.1. Nadzirano i nenadzirano učenje

Gledanje na učenje kao na proces „korištenja iskustva za stjecanje stručnosti ", nadzirano učenje opisuje scenarij u kojem "iskustvo", primjer treninga, sadrži značajne informacije koje nedostaju u neviđenim "primjerima ispitivanja" na koje se treba primijeniti naučeno znanje. U ovakvom okruženju, stečena stručnost ima za cilj predvidjeti nedostajuće informacije za testne podatke. U takvim slučajevima možemo o okolini razmišljati kao o učitelju koji "nadgleda" učenika pružajući dodatne informacije.

Varijable se mogu okarakterizirati kao kvantitativne ili kvalitativne (također poznate kao kategoričke). Kvantitativne varijable poprimaju numeričke vrijednosti. Primjeri takvih vrijednosti su starost, visina ili doprinos osobe, vrijednost kuće ili cijena dionice. Suprotno tome, kvalitativne varijable poprimaju vrijednosti u jednoj od K različitih klasa ili kategorija. Primjeri kvalitativnih varijabli klase uključuju spol osobe (muški ili ženski), marku kupljenog proizvoda (marka A, B ili C), neovisno ima li osoba dug (da ili ne) ili dijagnozu raka (Akutna mijelogena leukemija, Akutna Limfoblastična leukemija ili nema leukemije). Probleme s kvantitativnim odgovorom obično definiramo kao regresijski problem, dok one koji uključuju kvalitativne odgovore često nazivamo klasifikacijskim problemima. Međutim, razlika nije uvijek tako jasna i čista.

Linearna regresija najmanjeg kvadrata koristi se kvantitativnim odgovorima, dok se logistička regresija obično koristi s kvalitativnim (dvorazrednim ili binarnim) odgovorima. Kao takva, često se koristi kao metoda klasifikacije,ali budući da procjenjuje vjerojatnosti klase, također se može smatrati i regresijom. Neke statističke metode, poput K-najbližih susjeda (eng. *K-nearest neighbors*) i pojačavanje (eng. *boosting*), mogu se koristiti u slučaju bilo kvantitativnih ili kvalitativnih odgovora. Skloni smo odabiru statističkih metoda učenja na temelju toga da li je odgovor kvantitativan ili kvalitativan; to jest., mogli bismo koristiti linearnu regresiju kada je kvantitativan i logističku regresiju kada je kvalitativan. Međutim, jesu li prediktori kvalitativni ili kvantitativni, općenito se smatra manje važnim. Većina statističkih metoda učenja se može primijeniti prije analize bez obzira na ponuđeni tip varijable prediktora. [13]

U nenadziranom učenju, za razliku od nadziranog učenja, nema razlike između podataka treninga i podataka testiranja. Učenik obrađuje ulazne podatke s ciljem pronalaska sažete ili kompresirane verzije tih podataka. Grupiranje skupa podataka u podskupine sličnih predmeta tipičan je primjer takvog zadatka. [11]

4.3.2. Aktivni i pasivni učenici

Paradigme učenja mogu se razlikovati ovisno o ulozi koju igra učenik. Razlikujemo aktivne i pasivne učenike. Aktivni učenik komunicira s okolinom za vrijeme treninga, recimo, postavljanjem upita ili izvođenjem eksperimenata. Pasivni učenik, za razliku od aktivnog učenika, promatra samo informacije koje mu okolina ili učitelj pruža, bez utjecaja ili usmjeravanja. Vratimo se na primjer s neželjenom e-poštom. Treba primijetiti da je učenik filter za neželjenu e-poštu obično pasivan - čeka da korisnici označe e-poštu koja im dolazi. U aktivnom okruženju moglo bi se zamisliti da se od korisnika traži da označe određene e-mailove koje je učenik odabrao ili ih je čak sam sastavio kako bi poboljšao svoje razumijevanje oko toga što je to neželjena e-pošta. [11]

4.3.3. Korisnost učitelja

Kad se razmišlja o ljudskom učenju, kod bebe kod kuće ili učenika u školi, postupak često uključuje korisnu pomoć učitelja koji učenika pokušava nahraniti informacijama najkorisnijim za postizanje cilja učenja. Nasuprot tome, kad znanstvenik uči o prirodi, okoliš koji igra ulogu učitelja, u najboljem slučaju može biti smatran pasivnim učiteljem - jabuke padaju, zvijezde sjaje i kiša pada bez obzira na potrebe učenika. Takve scenarije učenja modeliramo pretpostavljajući da se podaci o treningu (ili iskustvu učenika) generiraju nekim slučajnim postupkom. Ovo je osnovni gradivni blok u grani "statističko učenje". Učenje se događa i kada doprinos učenika generira kontradiktorni "učitelj". Ovo može biti slučaj u primjeru filtriranja neželjene pošte (ako se pošiljatelj neželjene pošte potrudi da zavede dizajnera filtriranja neželjene pošte). Kontradiktorni model učitelja se također koristi kao najgori scenarij kada se ne može sigurno pretpostaviti blaže postavljanje. Ako možete učiti usprkos kontradiktornom učitelju, zajamčeno ćete uspjeti u interakciji s bilo kojim neobičnim učiteljem. [11]

5. Teorija vjerojatnosti

Da bismo znali baratati sa instancama u kojima se strojno učenje može koristiti, trebamo razviti odgovarajući jezik pomoću kojeg ćemo moći jezgrovito opisati probleme. U nastavku će se definirati određeni pojmovi vezani uz ovaj rad i dati poprilično neformalan pregled teorije vjerojatnosti. [9]

5.1. Slučajne varijable

Pretpostavimo da smo bacili kocku i htjeli bismo znati svoje šanse da se okrenulo 1, a ne neka druga znamenka. Ako su kockice poštene svih šest ishoda $X = \{1, ..., 6\}$ imaju jednaku vjerojatnost da se ostvare, stoga bismo približno dobili 1 u 1 od 6 slučajeva. Teorija vjerojatnosti omogućuje nam modeliranje nesigurnosti u ishodu takvih eksperimenata. Formalno navodimo da se 1 javlja s vjerojatnošću $\frac{1}{6}$. U mnogim eksperimentima, poput bacanja kocke, ishodi su numeričke prirode i s njima se lako možemo nositi. U drugim slučajevima, ishodi možda neće biti brojčani, npr. ako bacamo novčić i promatramo glave ili pisma. U u tim je slučajevima korisno numeričke vrijednosti pridružiti ishodima. Ovo se radi putem slučajne varijable. Recimo da slučajna varijabla X poprimi vrijednost $+1$ kad god se na novčiću okrene glava, a vrijednost -1 kada se okrene pismo. Za označavanje slučajnih varijabli X, Y koriste se velika slova, a malim slovima x, y se označavaju vrijednosti koje one poprimaju.

5.2. Distribucije

Možda najvažniji način karakteriziranja slučajne varijable je povezivanje vjerojatnosti s vrijednostima koje može poprimiti. Ako je slučajna varijabla diskretna, tj. poprima konačan broj vrijednosti, onda se ova dodjela vjerojatnosti naziva funkcija vjerojatnosti mase (eng. *probability mass function*) ili skraćeno PMF. PMF mora biti, po definiciji, nenegativan i suma mu mora biti jednaka 1. Na primjer, ako je novčić pošten, tj. glave i pisma su jednako vjerojatni, onda slučajna varijabla X poprima vrijednosti $+1$ i -1 s vjerojatnošću 0.5. To se može zapisati kao

$$Pr(X = +1) = 0.5$$

$$Pr(X = -1) = 0.5.$$

Pošto ne postoji opasnost od zabune (eng. *confusion*), upotrijebit ćemo pomalo neformalni zapis $p(x) := Pr(X = x)$.

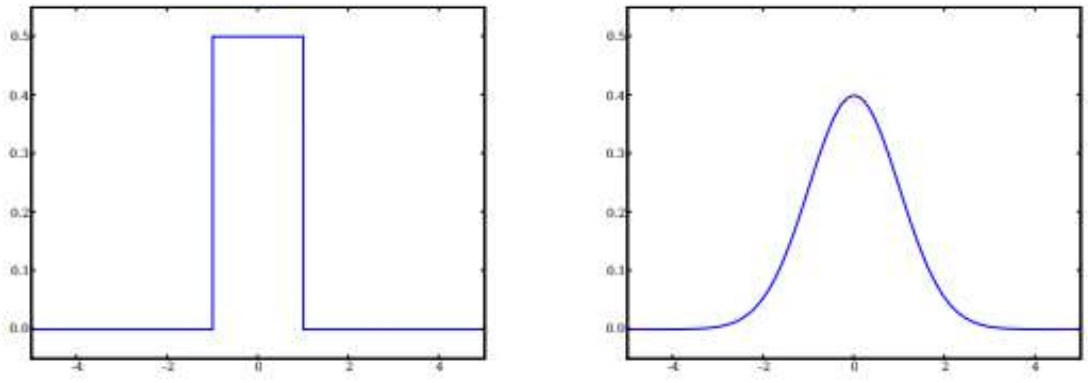
U slučaju kontinuirane slučajne varijable dodjeljivanje vjerojatnosti rezultira funkcijom gustoće vjerojatnosti (eng. *probability density function*) ili kraticom PDF. Uz neko zlostavljanje terminologije, ali u skladu s konvencijama, često ćemo koristiti gustoću ili raspodjelu umjesto funkcije gustoće vjerojatnosti. Kao u slučaju PMF, PDF također mora biti ne negativan i suma mu mora biti jednaka 1.

Slika 11 prikazuje jednoliku distribuciju (eng. *uniform distribution*)

$$p(x) = \begin{cases} \frac{1}{b-a}, & \text{ako je } x \in [a, b] \\ 0, & \text{inače} \end{cases}$$

i Gaussovu distribuciju (eng. *Gaussian distribution*) (također poznata pod nazivom normalna distribucija)

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$$



Slika 11: Jednolika distribucija u intervalu $[-1, 1]$ (lijevo). Gaussova distribucija s nultom srednjom i jediničnom varijancom (desno).(Izvor: [9], 2008)

5.3. Srednja vrijednost i varijanca

Uobičajeno pitanje o slučajnoj varijabli je ono što se očekuje da bi vrijednost mogla biti. Na primjer, kada mjerimo napon uređaja, možemo se zapitati koje bi to mogle biti njegove tipične vrijednosti. Kada odlučuje hoće li djetetu primijeniti hormon rasta, liječnik može pitati koliko bi trebao biti razuman raspon visine. U te svrhe moramo definirati očekivanja i povezane količine raspodjele.[9]

Srednja vrijednost je slučajna varijabla X koja se definira kao

$$\mathbb{E}[X] = \sum_x xp(x).$$

Na primjer, u slučaju kocke imamo jednake vjerojatnosti od 1/6 za svih 6 mogućih ishoda. Iz ovog jednostavnog promjera lako je vidjeti da je srednja vrijednost $(1+2+3+4+5+6)/6 = 3.5$.

Varijanca je slučajna varijabla X koja se definira kao

$$Var[X] := \mathbb{E}[(X - E[X])^2].$$

a ako je $f : \mathbb{R} \to \mathbb{R}$ funkcija, onda je varijanca od $f(X)$ zadana kao

$$Var[f(X)] := \mathbb{E}[(f(X) - E[f(X)])^2].$$

Varijanca mjeri za koliko prosječno $f(X)$ odstupa od očekivane vrijednosti. Gornja granica varijance može se koristiti za davanje garancije o vjerojatnosti da će $f(X)$ biti unutar svoje očekivane vrijednosti. To je jedan od razloga zašto je odstupanje često povezano s rizikom slučajne varijable. Često se raspravlja o svojstvima slučajne varijable u smislu njene standardne devijacije, koja je definirana kao kvadratni korijen varijance. [9]

5.4. Marginalizacija i neovisnost

S obzirom na dvije slučajne varijable X i Y, može se napisati njihova zajednička gustoća $p(x, y)$. S obzirom na zajedničku gustoću, integriranjem y se može izvaditi $p(x)$. Ta se operacija naziva marginalizacija:

$$p(x) = \int_y dp(x, y)$$

Ako je Y diskretna slučajna varijabla, tada integraciju možemo zamijeniti sumatorom

$$p(x) = \sum_y p(x, y)$$

Kažemo da su X i Y neovisni, to jest, da vrijednost koju poprima X ne ovisi o vrijednostima koje poprima Y kada

$$p(x, y) = p(x)p(y)$$

Neovisnost je korisna kada se radi s velikim brojem slučajnih varijabli čije ponašanje želimo zajednički procijeniti. [9]

5.5. Bayes-ovo pravilo

Pretpostavimo da su X i Y slučajne varijable. Tada vrijedi

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

To proizlazi iz činjenice da je $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$. Ključna posljedica gore navedene formule je da možemo preokrenuti uvjet između para slučajnih varijabli. [9]

5.6. Formalni model

Započnimo našu matematičku analizu i formalizaciju pokazujući koliko učenje može biti uspješno postignuto u relativno pojednostavljenom okruženju. Zamislite da ste tek stigli u neki mali pacifički otok. Ubrzo doznajete da su papaje važan sastojak u lokalnoj prehrani. Međutim, nikada prije niste kušali papaju. Morate naučiti kako predvidjeti je li papaja koju vidite na tržnici ukusna ili ne. Prvo, morate odlučiti na kojim osobinama papaje ćete temeljiti svoje predviđanje. Odlučujete na temelju vašeg prethodnog iskustva s drugim voćem koristiti dvije značajke: boju papaje (od tamnozelene do narančaste i crvene do tamno smeđe) i mekoću papaje (od tvrdo do mekano). Ulazni podaci za utvrđivanje vašeg pravila predviđanja je uzorak papaje gdje ste ispitali boju i mekoću, a zatim probali i saznali da li je plod bio ukusan ili ne. Analizirajmo ovaj zadatak kao demonstraciju razmatranja uključenu u probleme učenja.

Naš prvi korak je opisati formalni model čiji je cilj obuhvatiti takvo učenje zadataka. U osnovnom statističkom okruženju za učenje, učenik ima pristup sljedećem: [11]

- **Skup domene**: (eng. *domain set*) proizvoljan skup X. Ovo je skup objekata koje možda želimo označiti. Na primjer, u prethodno spomenutom problemu učenja odabira papaje, skup domene bit će skup svih papaja. Obično, ove točke domene bit će predstavljene vektorom značajki (eng. *features*), poput boje i mekoće papaje. Točke domene nazivamo i instancama (eng. *instances*) a X kao prostor instance (eng. *instance space*)

- **Skup oznaka:** (eng. *label set*). Za naše trenutne potrebe ograničit ćemo skup oznaka na dvo-elementni skup, koji se obično označava kao $0, 1$ ili $-1, +1$. Neka Y označava naš skup mogućih oznaka. Za naš primjer papaje, neka Y bude $0, 1$, gdje 1 predstavlja biti ukusan, a 0 znači biti neukusan.

- **Podaci za trening:** (eng. *training data*) $S = ((x_1, y_1)...(x_m, y_m))$ je konačan slijed parova u $X \times Y$: odnosno niz označenih točaka domene. Ovo su ulazni podaci kojima učenik ima pristup (poput niza papaja koje su kušane i njihove boje, mekoće i ukusa). Takvi označeni primjeri često se nazivaju primjeri treninga. S ponekad nazivamo i set za trening (eng. *training set*)

- **Izlaz učenika:** (eng. *learner's output*) od učenika se traži da iznese pravilo predviđanje, $h : X \to Y$. Ova se funkcija naziva i prediktor(eng. *predictor*), hipoteza(eng. *hypothesis*) ili klasifikator(eng. *classifier*). Prediktor se može koristiti za predviđanje oznaka novih točaka domene. U našem primjeru papaje, to je pravilo koje će naš učenik upotrijebiti za predviđanje hoće li buduće papaje koje ispituje na tržnici biti ukusno ili ne. Oznakom $A(S)$ označavamo hipotezu da algoritam učenja A vraća rezultat po primanju treniranog slijeda S.

- **Jednostavan model generiranja podataka:** (eng. *simple data-generation model*) Krenimo s objašnjavanjem kakvi se podaci generiraju u treningu. Prvo, pretpostavljamo da se instance (papaje koje susrećemo) generiraju nekom raspodjelom vjerojatnosti (u ovom slučaju predstavljajući okolinu). Označimo tu raspodjelu vjerojatnosti po X s D. Važno je napomenuti da ne pretpostavljamo da učenik zna bilo što o ovoj distribuciji. Za vrstu

zadataka učenja o kojima raspravljamo, ovo bi mogla biti bilo koja proizvoljna raspodjela vjerojatnosti. Što se tiče naljepnica, u trenutnoj raspravi pretpostavljamo da postoji neka "ispravna" funkcija označavanja, $f : X \to Y$, i da je $y_i = f(x_i)$ za sve i. Funkcija označavanja nepoznata je učeniku. Zapravo, to je upravo ono što učenik pokušava shvatiti. Ukratko, svaki se par u podacima iz treninga S generira prvim uzorkovanjem točke x_i prema D, a zatim ga označava s f.

- **Mjere uspjeha:** (eng. *measures of success*) Pogrešku klasifikatora definiramo kao vjerojatnost da ne predviđa ispravnu oznaku na generiranoj slučajnoj točki podataka gore spomenutom temeljnom distribucijom. Odnosno, pogreška h je vjerojatnost izvlačenja slučajne instance x, prema raspodjeli D, takva da $h(x) \neq f(x)$. Formalno, s obzirom na podskup domene $A \subset X$, raspodjela vjerojatnosti D dodjeljuje broj $D(A)$ koji određuje vjerojatnost promatranja točke $x \in A$. U mnogim slučajevima A nazivamo događajem (eng. *event*) i izražavamo ga pomoću funkcije $\pi : X \to \{0, 1\}$, $A = x \in X : \pi(x) = 1$. U tom slučaju, također koristimo oznaku $P_{x \sim D}[\pi(x)]$ kako bi izrazili $D(A)$.

  Pogrešku pravila predviđanja, $h : X \to Y$, definiramo kao

$$L_{D,f}(h) \stackrel{\text{def}}{=} \underset{x \sim D}{P}[h(x) \neq f(x)] \stackrel{\text{def}}{=} D(\{x : h(x) \neq f(x)\})$$

  Pogreška takvog h vjerojatnost je slučajnog odabira primjera x za koji je $h(x) \neq f(x)$. Indeks (D, f) ukazuje na to da se mjeri pogreška s obzirom na raspodjelu vjerojatnosti D i ispravnu funkciju označavanja f. Ovaj indeks se izostavlja kada je to jasno iz konteksta. $L_{(D,f)}(h)$ ima nekoliko sinonima kao što su pogreška generalizacije (eng. *generelisation error*), rizik (eng. *risk*) ili istinska pogreška (eng. *true error*) od h. Za pogrešku koristimo slovo L, budući da ovu pogrešku promatramo kao gubitak učenika.

Učenik ne vidi osnovnu distribuciju D i funkciju za označavanje f. U našem primjeru papaje, upravo smo stigli na novi otok a mi nemamo pojma o tome kako se papaje distribuiraju i kako predvidjeti njihovu ukusnost. Jedini način na koji učenik može komunicirati s okolinom je kroz promatranje seta treninga. [11]

5.7. Regresija

Zamislimo zadatak u kojem se želi pronaći jednostavan obrazac u podacima - funkcionalni odnos između X i Y komponenata podataka. Na primjer, želi se pronaći linearna funkcija koja najbolje predviđa djetetovu porođajnu težinu na temelju ultrazvučnih mjera opsega glave, opsega trbuha i duljine bedrene kosti. Naš skup se sastoji od domene X podskupa R^3 (tri ultrazvučna mjerenja) i skupa "naljepnica" Y, skupa stvarnih brojeva (težina u gramima). U ovom kontekstu, prikladnije je Y nazvati ciljnim skupom. Naši podaci o treningu kao i rezultati učenika su konačan niz parova (x, y) i funkcija od X do Y.

Kvalitetu funkcije hipoteze možemo procijeniti na sljedeći način, $h : X \to Y$, očekivanom kvadratnom razlikom između pravih i predviđenih oznaka vrijednosti, naime,

$$L_D(h) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{(x,y)\sim D}(h(x) - y)^2.$$

Da bismo se prilagodili širokom spektru zadataka učenja, generaliziramo ovaj formalizam mjere uspjeha na sljedeći način.

S obzirom na bilo koji skup H (koji igra ulogu naših hipoteza ili modela) i neke domena Z neka je δ bilo koja funkcija od $H \times Z$ do skupa nenegativnih realnih brojeva,

$\delta : H \times Z \to \mathbb{R}_+$.

Takve funkcije nazivamo funkcijama gubitka. Treba imati na da za probleme s predviđanjem imamo $Z = X \times Y$. Međutim, naš pojam funkcije gubitka generaliziran je izvan zadataka predviđanja, pa prema tome omogućuje Z da bude bilo koja domena primjera. Definirajmo sada funkciju rizika kao očekivani gubitak klasifikatora, $h \in H$, s obzirom na raspodjelu vjerojatnosti D nad Z, naime,

$$L_D(h) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{z\sim D}[\delta(h, z)]$$

.

Razmotrimo očekivanje gubitka h nad objektima z odabranim slučajno prema D. Slično tome, empirijski rizik definiramo kao očekivani gubitak nad danim uzorkom $S = (z1, ..., z_m) \in Z^m$, slijedi,

$$L_S(h) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m}\delta(h, z_i)$$

.

Funkcije gubitka korištene u prethodnim primjerima zadataka su sljedeće:

- **0-1 gubitak:** Ovdje se slučajna varijabla z kreće u skupu parova $X \times Y$ i funkcija gubitka je

$$\delta_{0-1}(h, (x, y)) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{ako je } h(x) = y \\ 1, & \text{ako je } h(x) \neq y \end{cases}$$

  Ova se funkcija gubitka koristi u binarnim ili višerazrednim problemima klasifikacije.

- **Kvadratni gubitak:** Ovdje se naša slučajna varijabla z kreće u skupu parova X $\times$ Y a funkcija gubitka je

$$\delta_{sq}(h, (x, y)) \stackrel{\text{def}}{=} (h(x) - y)^2$$

  Ova se funkcija gubitka koristi u problemima regresije. [11]

6. Klasifikacija

Klasifikacijski problemi se u praksi pojavljuju možda čak i češće nego regresijski problemi. Neki od primjera klasifikacijskoga problema su :

- Osoba dolazi na hitnu sa simptomima koji bi se mogli pripisati jednom od tri zdravstvena stanja. Koji od ta tri stanja pojedinac ima?
- Usluga internetskog bankarstva mora moći utvrditi da li je transakcija koja se provodi na web mjestu prijevara, temeljem korisnikove IP adrese, prošle povijesti transakcija i tako dalje.
- Na temelju podataka o sekvenci DNA za određeni broj bolesnika sa i bez određene bolesti, biolog bi želio otkriti koje su DNA mutacije štetne (uzrokuju bolesti), a koje nisu.

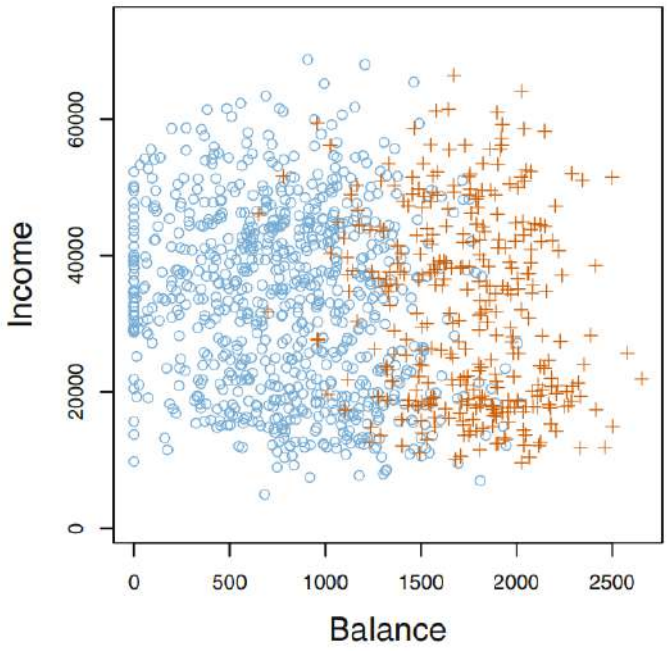
Kao i u regresiji, i u klasifikaciji (eng. *classification*)imamo skup opažanja iz treninga(eng. *training data*) $(x_1, y_1), ..., (x_n, y_n)$ koje možemo koristiti prilikom kreiranja klasifikatora. Želimo da naš klasifikator daje dobre rezultate ne samo na treniranim podacima, već i na testnim opažanjima koja se nisu koristila za vrijeme obuke klasifikatora.

Klasifikaciju ćemo ilustrirati pomoću simuliranog zadanog skupa podataka. Zanima nas predviđanje da li će pojedinac platiti kreditnom karticom na temelju godišnjih prihoda i mjesečnog stanja računa na kreditnoj kartici. Skup podataka je prikazan na slici 12 a sadrži godišnje prihode i mjesečno stanje računa kreditne kartice za podskup od 10 000 pojedinaca. Narančastom bojom su označeni pojedinci koji nisu podmirili svoje novčane obaveze u zadanom mjesecu, dok su plavom obojani oni koji jesu. Čini se da su pojedinci koji su podmirili svoje novčane obveze imali veći saldo na kreditnim karticama od onih koji nisu. Kreirati ćemo model za predviđanje zadane vrijednosti (Y) za bilo koju zadanu vrijednost bilance (X_1) i dohotka (X_2). Budući da Y nije kvantitativan, jednostavni linearni regresijski model ovdje nije primjenjiv. Vrijedno je za napomenuti da u ovom zadanom skupu podataka, prikazanom na slici 12, odnos između ravnoteže prediktora i zadane vrijednosti je vrlo izražen. U stvarnim aplikacijama klasifikacije, odnos neće biti tako jak, međutim, radi ilustracije klasifikacijskoga postupka, koristi se primjer u kojem je taj odnos donekle pretjeran. [13]

Pretpostavimo da tražimo procjenu f na temelju promatranja treninga $(x_1, y_1), ..., (x_n, y_n)$, gdje su sada $y_1, ..., y_n$ kvalitativni. Najčešći pristup za kvantificiranje točnosti procjene $\hat{f}$ je stopa pogrešaka u treningu, udio pogrešaka (eng. *error rate*) koje se naprave ako se stopa pogrešaka $\hat{f}$ primijeni na opservacije treninga.

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i).$$

Ovdje je $\hat{y}_i$ predviđena oznaka klase za i-to promatranje pomoću $\hat{f}$. $I(y_i \neq \hat{y}_i)$ je varijabla pokazatelja (eng. *indicator variable*) koja je jednaka 1 ako je $y_i \neq \hat{y}_i$ i 0 ako je $y_i = \hat{y}_i$. Ako je $I(y_i \neq \hat{y}_i) = 0$, onda je i-to opažanje pravilno klasificirano našom metodom klasifika-



Slika 12: Zadani skup podataka (Izvor: [13], 2017)

cije, inače je pogrešno klasificiran. Stoga gore navedena jednadžba izračunava udio netočnih klasifikacija.

Jednadžba 2.8 naziva se stopa pogreške u treningu (eng. *training error*) jer se izračunava na temelju podataka koji su korišteni za obuku našeg klasifikatora. Kao i u regresiji, najviše nas zanimaju stope pogrešaka koje proizlaze iz primjenjivanja našeg klasifikatora na testna opažanja koja nisu korištena u treningu. Stopa testne pogreške (eng. *test error rate*) povezana sa nizom testnih opažanja (x_0, y_0) dobivena je s

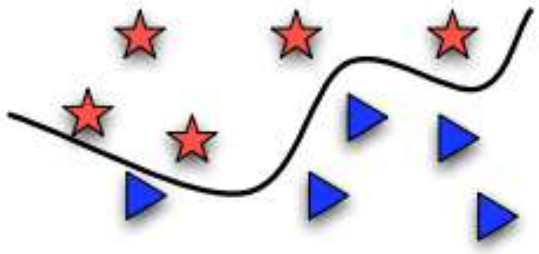
$$Ave(I(y_0 \neq \hat{y_0}))$$

gdje je $\hat{y_0}$ predviđena oznaka razreda koja proizlazi iz primjene klasifikatora na testno opažanje s prediktorom x_0. Dobar klasifikator je onaj za koji test greška daje najmanju vrijednost. [13]

6.1. Binarna klasifikacija

Binarna klasifikacija je vjerojatno najčešće proučavani problem u strojnom učenju. Tijekom prošlog stoljeća dovela je do velikog broja važnih algoritamskih i teoretskih razvoja. U svom najjednostavnijem obliku svodi na sljedeće: *S obzirom na uzorak x iz domene X, procijenite koju će vrijednost poprimiti pridružena binarna slučajna varijabla $y \in \{\pm 1\}$* . Ukratko, zadaci binarne klasifikacije uključuju jednu klasu koja ima normalno stanje i drugu klasu koja je abnormalno stanje. Klasi sa normalnim stanjem dodjeljuje se oznaka 0, a klasi s abnormalnim stanjem 1. Pošto ćemo rješenje za prepoznavanje bolesti palmine pipe kreirati uz pomoć binarne klasifikacije, bitno je da razumijemo ovaj problem. U našem zadatku, klase bi mogli

podijeliti na zdrave (normalno stanje) i zaražene (abnormalno stanje). [14]



Slika 13: Primjer binarne klasifikacije zvijezda i trokuta (Izvor: [9], 2008)

Za lakše razumijevanje, problem ćemo dodatno pojasniti primjerima. S obzirom na slike jabuka i naranči, možda bismo htjeli znati da li se na zadanoj slici nalazi jabuka ili naranča. Jednako tako bismo vjerojatno htjeli predvidjeti možemo li kao vlasnik kuće platiti zajam, s obzirom na podatke o prihodu i kreditnu povijest ili možda želimo provjeriti da li je neki e-mail neželjena ili željena pošta. Možemo uočiti da nam sposobnost rješavanja ovakvih osnovnih problema već omogućuje rješavanje velike raznolikosti praktičnih postavki. [9]

6.2. Popularni algoritmi

Sekciju u kojoj smo dali uvod u strojno učenje završiti ćemo opisom tri jednostavna algoritma (naivni Bayes, najbliži susjedi i logistička regresija) koji se mogu koristiti za rješavanje problema binarne klasifikacije. Svi navedeni algoritmi su lako upotrebljivi i jednostavno se implementiraju od nule u njihov najosnovniji oblik.

6.2.1. Naivni Bayes

Naivni Bayes (eng. *Naive Bayes*) klasična je demonstracija kako generativne pretpostavke i procjene parametara pojednostavljuju proces učenja. Pogledajmo problem predviđanja oznake $y \in \{0, 1\}$ na temelju vektora značajki $x = (x_1, ..., x_d)$, gdje pretpostavljamo da se svaki x_i nalazi u $\{0, 1\}$.

Za opis funkcije vjerojatnosti $P[Y = y|X = x]$ treba nam 2^d parametara, od kojih svaki odgovara $P[Y = 1|X = x]$ za određenu vrijednost $x \in \{0, 1\}^d$. To implicira da broj primjera koji nam trebaju, raste eksponencijalno s brojem značajki. U pristupu naivnog Bayesa iznosimo (prilično naivnu) generativnu pretpostavku da su značajke neovisne jedna o drugoj s obzirom na oznaku. To jest:

$$P[X = x|Y = y] = \prod_{i=1}^{d} P[Xi = xi|Y = y]$$

Uz ovu pretpostavku, i koristeći Bayesovo pravilo, Bayesov optimalni klasifikator može se dodatno pojednostaviti:

$$h_{Bayes}(x) = \underset{y\in\{0,1\}}{argmax} P[Y = y|X = x]$$

$$= \underset{y\in\{0,1\}}{argmax} P[Y = y]P[X = x|Y = y]$$

$$= \underset{y\in\{0,1\}}{argmax} P[Y = y] \prod_{i=1}^{d} P[Xi = xi|Y = y]$$

Prema tome, sada je broj parametara koje trebamo procijeniti samo 2d + 1. Ovdje je generativna pretpostavka značajno smanjila broj parametara koje moramo naučiti. Kada procijenimo parametre koristeći načelo najveće vjerojatnosti, dobiveni klasifikator naziva se klasifikator naivnog Bayesa (eng. *Naive Bayes classifier*). [11]

6.2.2. Najbliži susjedi

Algoritam najbližih susjeda jedan je od najjednostavnijih algoritama strojnog učenja. Ideja je zapamtiti set treninga, a zatim predvidjeti oznaku bilo koje nove instance na temelju oznaka njezinih najbližih susjeda iz seta treninga. Obrazloženje iza ove metode temelji se na pretpostavci da su značajke koje se koriste za opisivanje točaka domene relevantne za označavanja na način da susjedne točke vrlo vjerojatno imaju istu oznaku.

Pretpostavimo da je domena instance X obdarena metričkom funkcijom ρ. Odnosno, $\rho : X \times X \to \mathbb{R}$ je funkcija koja se vraća udaljenost između bilo koja dva elementa X. Neka je $S = (x_1, y_1), ..., (x_m, y_m)$ je slijed primjera treninga. Za svaki $x \in X$, neka je $\pi_1(x), ..., \pi_m(x)$ preuređivanje $\{1, ..., m\}$ prema njegovim udaljenostima do $x, \rho(x, x_i)$, odnosno, za sve $i < m$,

$$\rho(x, x_{\pi_i(x)}) \leq \rho(x, x_{\pi_{i+1}(x)}).$$

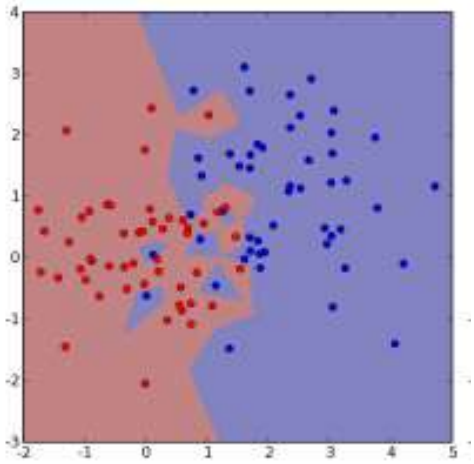
Za broj *k*, pravilo *k*-najbližeg susjeda za binarnu klasifikaciju definiramo na sljedeći način:

- **ulaz:** uzorak treninga $S = (x_1, y_1), ..., (x_m, y_m)$
- **izlaz:** za svaku točku $x \in X$, vrati oznaku većine među $\{y_{\pi_i(x)} : i \leq k\}$

Kad je $k = 1$, dobivamo pravilo 1-najbližeg susjeda: $h_S(x) = y_{\pi_1(x)}$. Granice dobivene klasifikacijom 1-najbližeg susjeda prikazane su na slici 14. [11]

6.2.3. Logistička regresija

Kao što smo već naveli u sekciji Regresija, logistička regresija se koristi kada imamo kvalitativne (binarne) odgovore i kao takva, često se koristi kao metoda klasifikacije. $h(x)$ možemo protumačiti kao vjerojatnost da je oznaka od x 1. Klasa hipoteze povezana s logističkom regresijom je kompozicija sigmoidne funkcije $\phi sig : \mathbb{R} \to [0, 1]$ sa razredom linearnih funkcija L_d. Konkretno, sigmoidna funkcija koja se koristi u logističkoj regresiji je logistička funkcija,



Slika 14: Granice dobivene iz klasifikatora 1-najbližeg susjeda (Izvor: [9], 2008)

definirana kao

$$\phi_{sig}(z) = \frac{1}{1 + exp(-z)}$$

Hipoteza je stoga (gdje se radi jednostavnosti koriste homogeno linearne funkcije):

$$H_{sig} = \phi_{sig} \circ L_d = \{x \mapsto \phi_{sig}(\langle w, x \rangle) : w \in \mathbb{R}^d\}$$

Valja primijetiti da je kad je $\langle w, x \rangle$ vrlo velik, tada $\phi_{sig}(\langle w, x \rangle)$ teži prema 1, a dok je $\langle w, x \rangle$ vrlo malen tada $\phi_{sig}(\langle w, x \rangle)$ teži prema 0.

Sljedeće bi trebali definirati funkciju gubitaka. Ako uzmemo u obzir set treninga $S = (x_1, y_1), ..., (x_m, y_m)$, problem empirijskog minimiziranja rizika koji je povezan s logističkom regresijom glasi

$$\underset{w \in \mathbb{R}^d}{argmin} \frac{1}{m} \sum_{i=1}^{m} log(1 + exp(-y_i \langle w, x_i \rangle)).$$

Prednost funkcije logističkog gubitka je ta što je ona konveksna funkcija koja uzima w kao neovisnu varijablu; stoga se empirijsko minimiziranje rizika može učinkovito riješiti pomoću standardne metode. [11]

7. Neuronske mreže

Neuronske mreže (eng. *neural networks*) jedna su od najljepših programskih paradigmi ikad izmišljenih. U konvencionalnom pristupu programiranju, govorimo računalu što treba učiniti tako da mu razlomimo probleme na mnogo malih, precizno definiranih zadatka koje računalo može izvršiti s lakoćom. Suprotno tome, u neuronskoj mreži ne govorimo računalu kako riješiti naš problem. Umjesto toga, on uči na temelju podataka promatranja, pronalazeći vlastito rješenje problema. Automatsko učenje iz podataka zvuči obećavajuće, međutim, do 2006. nismo znali kako osposobiti neuronske mreže da nadmašuju tradicionalnije pristupe, osim nekoliko usko specijaliziranih problema. Ono što se promijenilo 2006. godine bilo je otkriće tehnika za učenje u takozvanim dubokim neuronskim mrežama. Te su tehnike danas poznate kao duboko učenje (eng. *deep learning*) . S vremenom su se sve više razvijale, a danas duboke neuronske mreže i duboko učenje postižu izvanredne performanse na mnogim važnim problemima u računalnom vidu, prepoznavanju govora i obradi prirodnog jezika. [15]

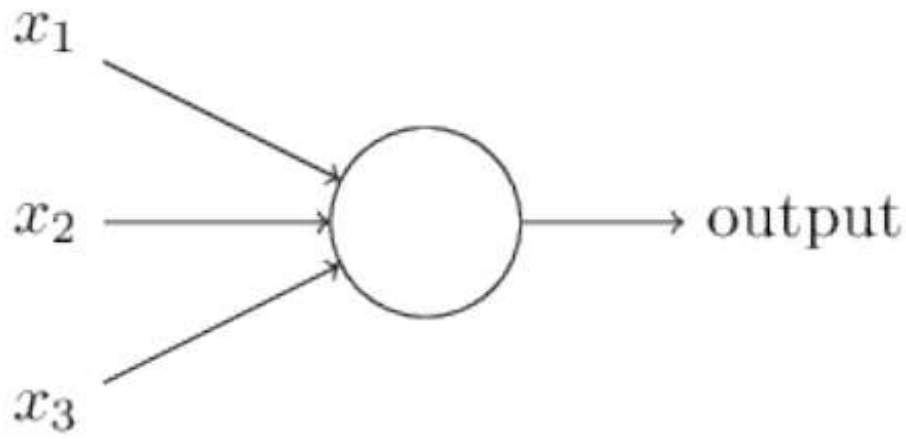
Neuronske mreže su nastale inspirirane učenjem stvarnih neurona u biološkom svijetu. Sastoje se od međusobno povezanih neurona koji kroz cijelu neuronsku mrežu propagiraju signale koje se podešavaju u odnosu na težine veza između neurona i aktivacijske funkcije neurona. Aktivacijske funkcije se koriste za izračunavanje signala prije nego se signal proslijedi dalje u mrežu. Tipovi aktivacijskih funkcija su:

- Linearna funkcija: $f(x) = x$
- Sigmoidna funkcija: $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$
- Funkcija tangensa hiperboličnog: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- ReLu: $f(x) = \begin{cases} 0, & \text{ako je } x < 0 \\ x, & \text{ako je } x \geq 0 \end{cases}$
- Softmax funkcija: $f(x_i) = \frac{e^{x_i}}{\sum_j ex_j}$

7.1. Perceptron

Osnovna gradivna jedinica umjetnih neuronskih mreža je umjetni neuron. Perceptron je umjetni neuron. Perceptrone je 1950-ih i 1960-ih razvio znanstvenik Frank Rosenblatt, nadahnut ranijim radovima Warrena McCullocha i Waltera Pittsa. Danas je uobičajenija upotreba drugih modela umjetnih neurona - u ovom radu i u mnogim modernim radovima s neuronskim mrežama, glavni neuronski model koji se koristi naziva se sigmoidni neuron. Da bismo razumjeli zašto su sigmoidni neuroni definirani na način na kakav jesu, vrijedi odvojiti malo vremena za razumijevanje perceptrona.

Perceptron uzima nekoliko binarnih ulaza, $x_1, x_2, ...$, i proizvodi jedan binarni izlaz. U prikazanom primjeru na Slici 15 perceptron ima tri ulaza x_1, x_2, x_3. Općenito bi mogao imati



Slika 15: Perceptron (Izvor: [15], 2018)

više ili manje ulaza. Rosenblatt je predložio jednostavno pravilo za izračunavanje rezultata. Uveo je težine, $w_1, w_2, ...$, stvarne brojeve koji izražavaju važnost pojedinog ulaza na izlaz.

Izlaz neurona, 0 ili 1, određuje se tako da je gleda da li je ponderirana suma $\sum_j w_j x_j$ manja ili veća od nekog praga. Baš kao i težina, prag je stvaran broj koji je parametar neurona. Ovo sve bi se algebarski moglo prikazati kao:

$$\text{izlaz} = \begin{cases} 0, & \text{ako je } \sum_j w_j x_j \leq \text{prag} \\ 1, & \text{ako je } \sum_j w_j x_j > \text{prag} \end{cases}$$

7.2. Sigmoidni neuron

Sigmoidni neuroni slični su perceptronima, ali modificirani tako da male promjene u njihovoj težini i pristranosti uzrokuju samo malu promjenu u njihovom izlazu. To je presudna činjenica koja će omogućiti mreži sigmoidnih neurona da uči.

Baš kao i perceptron, sigmoidni neuron ima ulaze $x_1, x_2, ...$ Ali umjesto da je ulaz samo 0 ili 1, ti ulazi mogu poprimiti bilo koje vrijednosti između 0 i 1. Dakle, na primjer, 0,638. . . je valjani ulaz za sigmoidni neuron. Isto kao i perceptron, sigmoidni neuron ima težinu za svaki ulaz, $w_1, w_2, ...$, i ukupnu pristranost b. Ali izlaz nije 0 ili 1. Umjesto toga, on je $\sigma(wx + b)$, gdje se σ naziva sigmoidna funkcija, a definirana je sa:

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

Da se izrazimo sve malo eksplicitnije, izlaz sigmoidnog neurona s ulazima $x_1, x_2, ...$, težinama $w_1, w_2, ...$, i pristranosti b je

$$\frac{1}{1 + exp(-\sum_j w_j x_j - b)}$$

8. Implementacija aplikacije

U ovom poglavlju ćemo svo dosad navedeno teorijsko znanje iskoristiti za izradu aplikacije. Objasniti ćemo razvoj aplikacije od njezinih početnih faza planiranja, prikupljanja i transformacije podataka, modeliranja i implementacije pa sve do testiranja.

8.1. Podaci

Kao i u svakom projektu strojnog učenja, najvažniji su kvalitativni i kvantitativni podaci. Iako je problem koji se opisuje u ovom radu dosta istraživan i razvijeno je nekoliko metoda detekcije palmine pipe. Unatoč tome, eksperimentalna metoda prepoznavanja bolesti palmine pipe snimanjem krune palme iz zraka pomoću bespilotne letjelice nije još istražena. Pošto je ova metoda jedna od pionira u tom području, podataka (fotografije palmi iz zraka) gotovo da nije ni bilo. Većina dostupnih fotografija je bilo snimanih iz žablje perspektive. Zbog manjka dostupnih podataka, odlučili smo se za samostalno prikupljanje podataka.

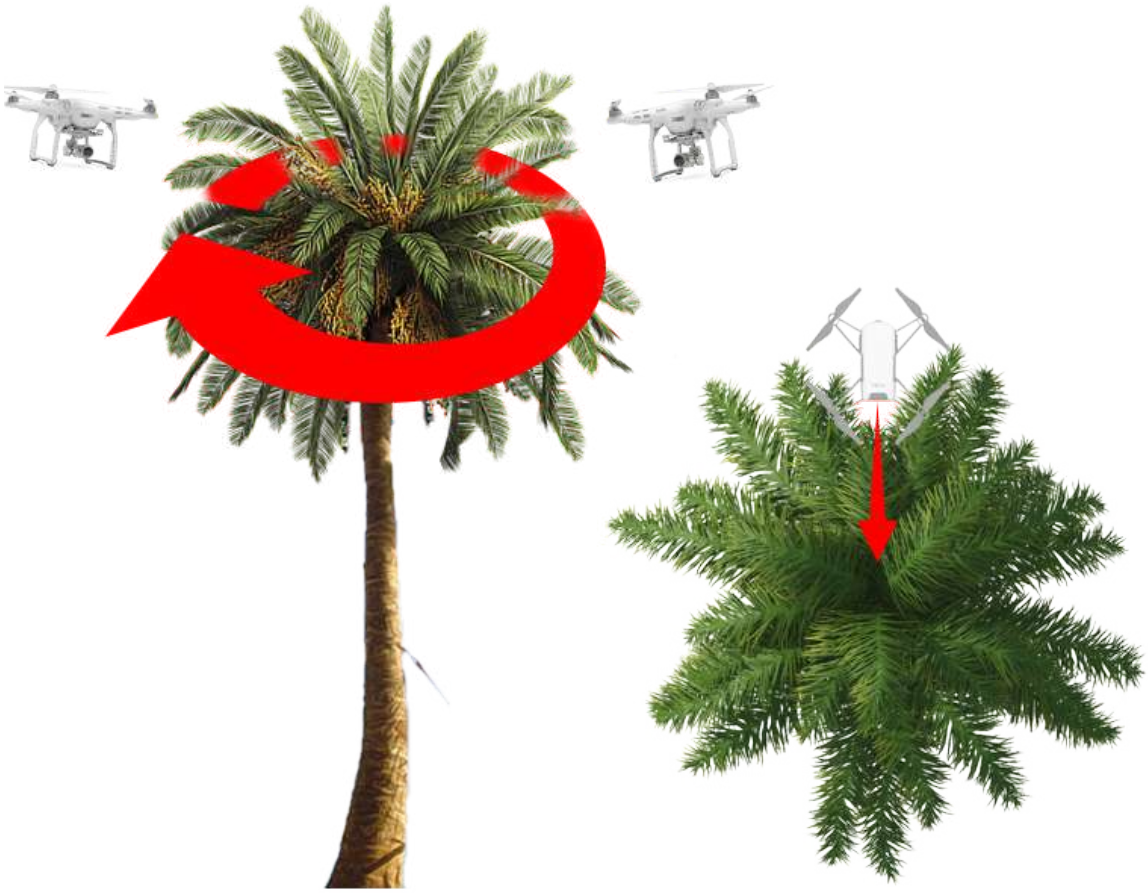
8.1.1. Prikupljanje podataka

Slika 16: Stručni suradnik mag.ing.agr. Luka Čotić (Izrada autora, 2020)

Količina prikupljenih podatka je dosta manja od planirane. Najveći nam je problem bio COVID-19 pošto smo najviše snimanja planirali odraditi u vrijeme proljeća kada se palme obilaze i tretiraju preparatima protiv palmine pipe. Još jedan od problema s kojima smo se suočili su bile dozvole za letenje bespilotnom letjelicom po rivama i šetnicama uz obalu. Većina se palmi nalazi na rivama i šetnicama uz obalu, ali nije bilo lako dobiti dozvole za letenje

iznad tih objekata iako bi se letovi vršili u svrhu zaštite palmi. Gradovima se nije svidjela ideja ometanja posjetitelja tijekom sezone. Palme sa splitske rive snimljene su iz auto košara kako bi se mogla snimiti kruna palme. Jedna od ljepših lokacija na kojima smo snimali bila je Kaštel Gomilica. U Kaštel Gomilici smo na raspolaganju imali preko četrdeset zdravih i zaraženih palmi u drvoredu, a na toj lokaciji smo prikupili većinu podataka za učenje modela. Kod prikupljanja podataka pomogao nam je stručni suradnik mag.ing.agr. Luka Čotić koji nas je vodio na lokacije i pomogao nam sa klasifikacijom zdravih i bolesnih palmi.

Podatke smo prikupljali na način da smo bespilotnom letjelicom okružili palmu i na kraju joj snimili krunu iz zraka kako bi se vidjela forma cijele krune. Na slici 17 može se vidjeti dijagram koji pobliže prikazuje proces snimanja pojedine palme, tj., pozicioniranja bespilotne letjelice i njezine kamere naspram palme. Svi podaci su se prvobitno prikupljali kao videozapisi. Vremenski je, u prosjeku, bilo potrebno oko 100 sekundi za snimanje pojedine palme. Vrijeme snimanja palme je ovisilo o promjeru palmine krune i visine palme.



Slika 17: Prikaz kutova snimanja svake pojedine palme (Izrada autora, 2020)

Komercijalne bespilotne letjelice su relativno nova tehnologija i iako imaju puno prednosti, imaju i neke nedostatke. Najveći nedostatak bespilotnih letjelica je duljina trajanja baterije. Kapacitet baterije bespilotne letjelice koju smo koristili za snimanje palmi bio je 4480 mAh, što nam je omogućilo oko 15 minuta leta sa snimanjem. Vrijeme punjenja baterije nakon 15 minuta leta i snimanja iznosila je oko 80 minuta. Kako ne bi trošili cijeli dan za snimanje par palmi, koristili smo ukupno tri baterije koje smo ciklički punili i izmjenjivali. Snimke snimljene

kamerom bespilotne letjelice, pohranjivale su se na MicroSD karticu koja je bila umetnuta u kameru. Prilikom snimanja, snimke su se pohranjivale i na mobitel preko kojeg se upravljala letjelica, ali te su snimke bile značajno manje kvalitete od onih pohranjenih na MicroSD kartici.

8.1.2. Transformacija podataka

Podaci su se prikupljali u obliku videozapisa. Za treniranje modela su nam potrebne fotografije. Kako bismo mogli trenirati model, vadili smo fotografije iz videozapisa i pohranjivali ih za daljnju obradu. Kamera koju smo koristili za snimanje videozapisa snimala je po 30 fps (eng. *frames per second*). Broj sličica u sekundi, izražen kao fps i predstavlja broj sličica koje kamera može snimiti u sekundi. Prema ovoj definiciji, naša kamera je snimala 30 sličica u sekundi. 30 fps je trenutni industrijski standard za jasan i gladak videozapis.

Što je broj sličica veći, to će videozapis biti glatkiji, a niže brzine mogu rezultirati isprekidanim ili slomljenim kretanjem snimanih objekata. Brzina sličica također utječe na veličinu video datoteka. Veća brzina kadrova od 60 fps rezultira s više sličica, pa će video datoteke biti veće. 30 fps-a se u našem slučaju pokazalo kao idealni broj sličica u sekundi. [16]

Nakon što smo snimili palme, videozapise smo s kartice prebacili na računalo gdje smo dalje upravljali s njima i transformirali ih. Programsko rješenje za kreiranje fotografija iz videozapisa kreirano je u Python programskom jeziku uz pomoć OpenCV i os biblioteka, a programski kod se može vidjeti u nastavku.

```
import cv2
import os
# citanje videa sa zadane pitanje
cam = cv2.VideoCapture("G:\\Palme\\VideoSnimke\\100MEDIA\\DJI_0044.MP4")

try:
        # kreiranje datoteke naziva data
        if not os.path.exists('data'):
                os.makedirs('data')
# iako ne postoji baci error
except OSError:
        print ('Error: Creating directory of data')

currentframe = 0
skipFrame = 30

while(True):
        # citanje video okvira(frame)
        ret,frame = cam.read()
        # ako video jos traje nastavi kreirati slike

        if ret:
                # procitaj svaki n-ti okvir (zadano sa skipFrame)
                if(currentframe % skipFrame == 0):
                        #name = './data/zdrava_17_' + str(int(currentframe/skipFrame
                            )) + '.jpg'
                        name = './data/bolensa_24_' + str(int(currentframe/skipFrame
```

```
                        )) + '.jpg'
                    print ('Creating...' + name)

                    # zapisi procitanu sliku
                    cv2.imwrite(name, frame)

            currentframe += 1
    else:
            break

# Oslobodi sav prostor i prozore nakon zavrsetka
cam.release()
cv2.destroyAllWindows()
```

Kako bismo uhvatili videozapis, moramo stvoriti objekt VideoCapture. Njegov argument može biti indeks uređaja (za prijenos uživo) ili naziv video datoteke (za prethodno snimljene video datoteke). Nakon što smo uhvatili videozapis preko putanje, provjeravamo da li postoji direktorij u koji želimo pohranjivati podatke koje ćemo kasnije kreirati. Direktorij smo nazvali data i ako data direktorij ne postoji, kreiraj ga.

Varijabla currentFrame nam označava trenutnu sličicu iz videa. Varijabla skipFrame nam služi za preskakanje određenog broja sličica. Ovdje smo inicijalizirali skipFrame na 30 iz razloga što nam je kamera snimala videozapise u 30 fps. Definiranjem varijable skipFrame na 30, preskočit ćemo svakih 30 sličica, što znači da ćemo preuzeti 1 sličicu po 1 sekundi videozapisa.

Naredba cam.read() dohvaća, dekodira i vraća sljedeći video okvir (sličicu). Ako postoji sljedeći video okvir, što znali da video još uvijek traje, čitamo svaki 30-ti okvir i pohranjujemo ga pod definiranim nazivom pomoću cv2.imwrite() naredbe. Nakon što smo prošli kroz sve video okvire videozapisa, video je završio, oslobađamo sav prostor i prozore prije zatvaranja programa.

Nakon prolaska kroz sve videozapise i preuzimanja video okvira iz njih, dobili smo skup podataka od ukupno 2724 fotografija. Kako je to i dalje jako malen broj podataka za učenje, iskoristili smo neke od značajki Keras-a za povećanje ulaznih podataka. Funkcija koja nam omogućuje generiranje serije podataka tenzorskih slika uz povećanje podataka u stvarnom vremenu zove se ImageDataGenerator(). Podaci će se vrtjeti u petlji (u skupinama). Iz ImageDataGenerator smo koristili sljedeće argumente:

- **rescale:** faktor skaliranja. Zadana vrijednost je None. Ako je None ili 0, neće se primijeniti ponovno skaliranje, u suprotnom množimo podatke s navedenom vrijednošću (nakon primjene svih ostalih transformacija)
- **shear_range:** Float vrijednost. Intenzitet pomicanja (Kut pomicanja u smjeru suprotnom od kazaljke na satu u stupnjevima)
- **zoom_range:** Float ili [donji, gornji] vrijednost. Raspon slučajnog zumiranja
- **vertical_flip:** Boolean vrijednost. Nasumično vertikalno zrcaljenje ulaznih podataka

Kako generiranje i augmentacija podataka izgleda u kodu i koje su vrijednosti korištene kod određenih atributa, može se vidjeti u sljedećem bloku koda. Iz priloženog koda, možemo vidjeti da se augmentacija podataka koristila i za podatke treniranja i za podatke verifikacije, tj. testiranja.

```
# create data generator
 # this is the augmentation configuration we will use for training
 train_datagen = ImageDataGenerator(
     rescale=1. / 255,
     shear_range=0.2,
     zoom_range=0.2,
     vertical_flip=True)

 # this is the augmentation configuration we will use for testing:
 test_datagen = ImageDataGenerator(
     rescale=1. / 255,
     vertical_flip=True)
```

Prije modeliranja, sve podatke je potrebno pravilno strukturirati. Podaci su podijeljeni u dva direktorija naziva train i validation. Svaki od tih direktorija u sebi sadrži po još dva direktorija koji predstavljaju nazive dvije klasa; bolesne i zdrave. U direktoriju bolesne nalaze se podaci (fotografije) zaraženih palmi, dok se u direktoriju zdrave nalaze podaci (fotografije) zdravih palmi. Skup podataka stukturiran je na sljedeći način:

- train
  - bolesne
    - bolesna1.png
    - bolesna2.png
    - ...
  - zdrave
    - zdrava1.png
    - zdrava2.png
    - ...
- validation
  - bolesne
    - bolesna1.png
    - bolesna2.png
    - ...
  - zdrave
    - zdrava1.png
    - zdrava2.png
    - ...

8.2. Model

Kreiranje dobrog modela nije lako. Rijetko kada će prvi testirani model biti onaj najbolji, pa je potrebno odraditi veliki broj iteracija početnog modela, ili kreirati potpuno novi. Zbog ograničenja sa GPU-ovom procesnom snagom, proces testiranja kreiranog modela bio je iznimno spor. Kako ne bismo uništili GPU i radnu memoriju zbog pregrijavanja, kreirali smo i testirali između 3 do 5 modela dnevno.

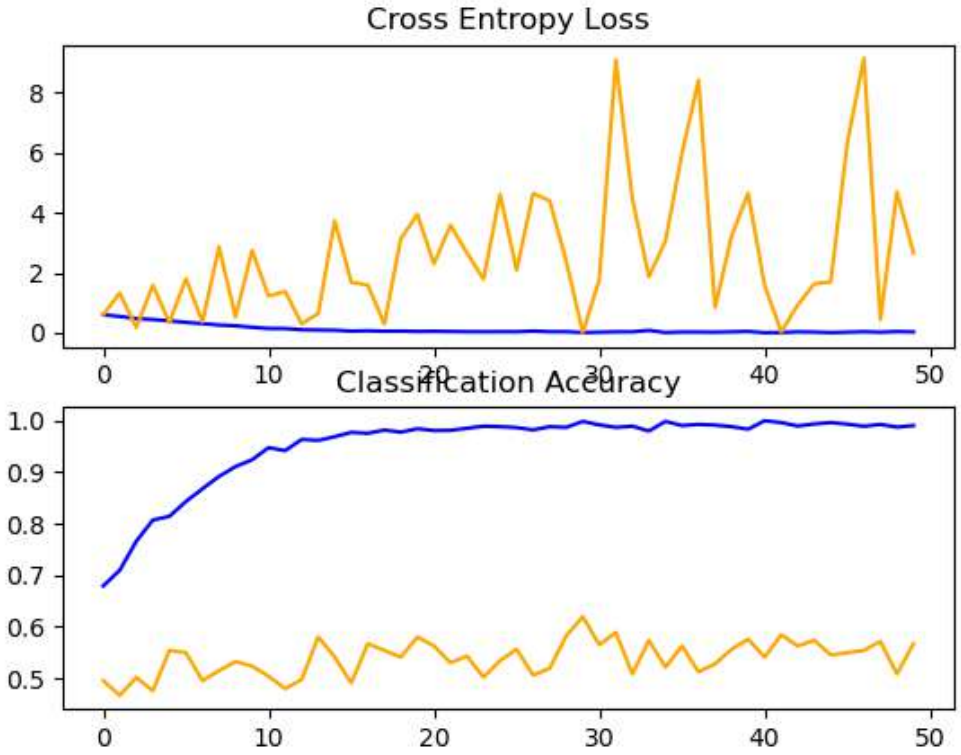
8.2.1. Kreiranje i testiranje

Prvi model koji smo kreirali nije baš najbolje ispao, što se može vidjeti i prema grafu sa Slike 18. Model sa slike je tipičan primjer pretreniranosti (eng. *overfitting*).Problem pretreniranosti jedan je od najvećih problema kod učenja modela. Problem je u tome što model počinje previše odgovarati konkretnim primjerima na kojima je treniran, čime više ne odgovara generalnim uzorcima već samo tim specifičnima iz skupa za učenje. Ukratko, model neviđene podatke ne generalizira dobro na temelju podataka iz treninga. Naš model predviđa ishode podatka iz treninga s točnosti od 99%, što je odličan rezultat, ali, ako mu se daju novi, neviđeni podaci, njihove ishode predviđa s točnosti od samo 53%.

Isto tako se može vidjeti da postoje ogromni skokovi entropijskih gubitaka kod validacijskih podataka. Iako bi povećanjem broja ulaznih podataka vjerojatno riješili ovaj problem i dobili poprilično dobar model, to nam nažalost nije jedna od opcija, pa je potrebno proučiti model i pokušati nekako riješiti ovaj problem. Pretpostavljamo da je razlog takvog ponašanja premali broj grupa (eng. *batch size*). Povećanjem količine grupa i micanjem Dropout() funkcije, taj bi se problem trebao riješiti, tj. trebao bi se smanjiti raspon između susjednih entropijskih gubitaka.

```
epochs = 50
batch_size = 5

def define_model():
    if K.image_data_format() == 'channels_first':
        input_shape = (3, img_width, img_height)
    else:
        input_shape = (img_width, img_height, 3)

    model = Sequential()
    model.add(Conv2D(16, (3, 3), input_shape=input_shape, activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Droput(0.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(1, activation='relu'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
```

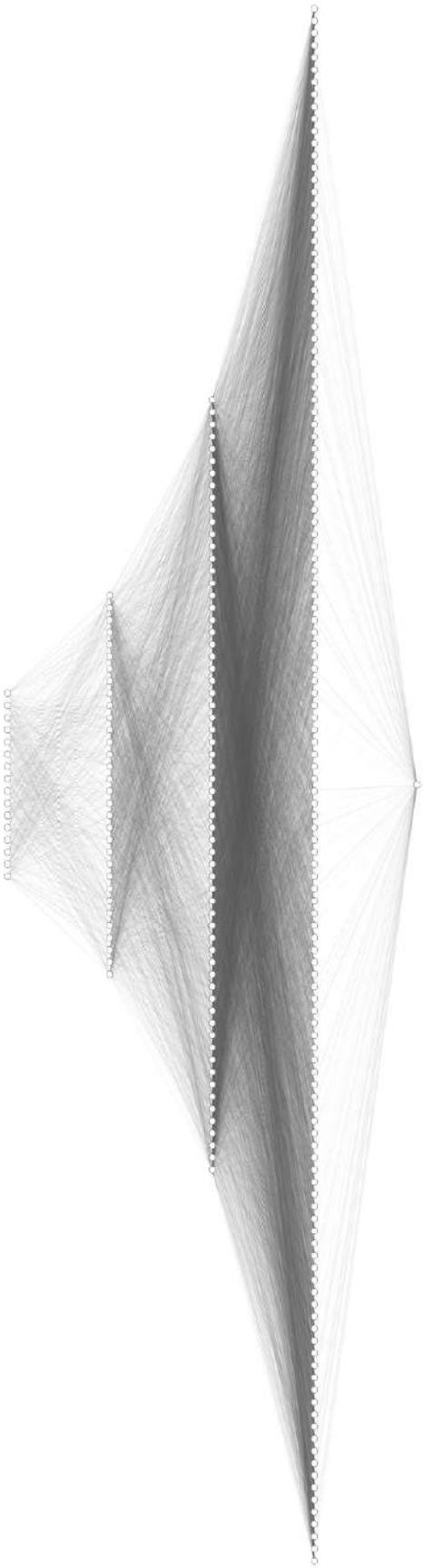
```
                metrics=['accuracy'])
return model
```



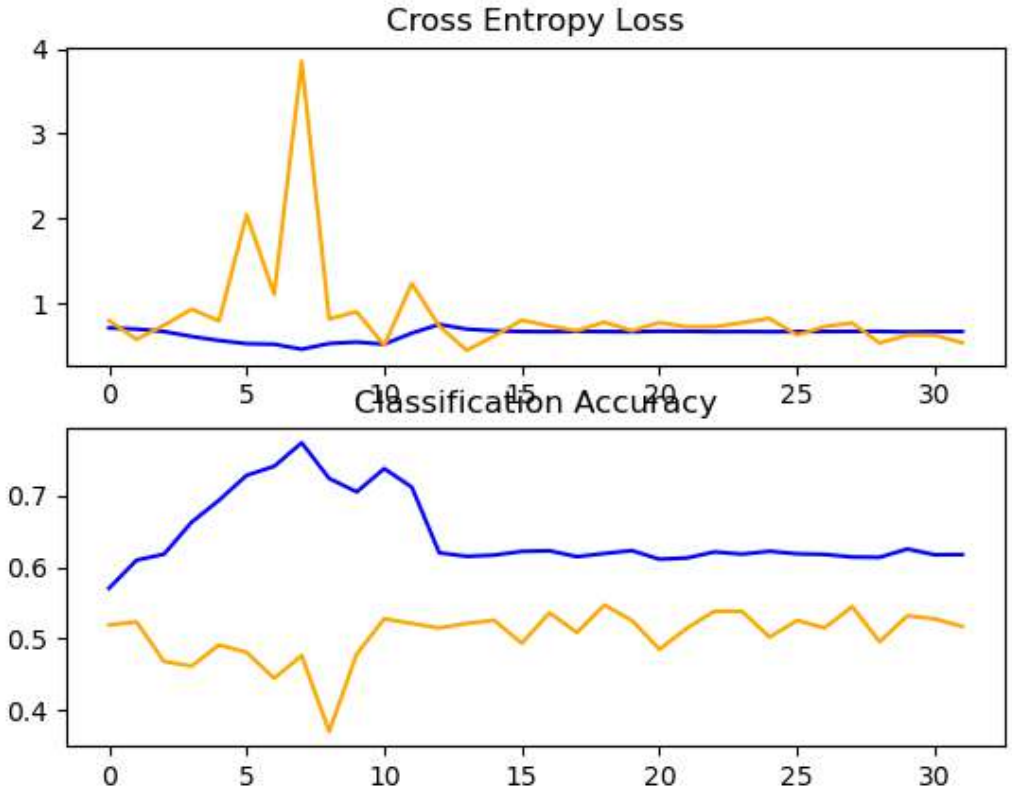
Slika 18: Unakrsni gubitak entropije i točnost klasifikacije modela 1; plava - trenirani podaci, narančasta - testirani podaci (Izrada autora, 2020)

Prema isječku bloka koda i prema Slici 19 možemo vidjeti da je model kojeg koristimo sekvencijalni. Sekvencijalni model se sastoji od 16 ulaza i 1 izlaza. Model također sadrži i 3 sakrivena sloja. Optimizer koji se koristio kod učenja je "adam" kojemu je zadana stopa učenja 0.001. Binary_crossentropy je korištena kao funkcija gubitaka pošto imamo samo dvije klase (zdrava i bolesna), a metrika koja je korištena za procjenu performanski našeg modela je "accuracy".

Povećanjem količine grupa, smanjenjem broja epoha i micanjem Dropout funkcije() smanjili smo problem pretreniranja, dok nam je točnost predviđanja ishoda ostala gotovo nepromjenjena. U ovom modelu, točnost predviđanja ishoda na treniranim podacima iznosi 61%, dok točnost predviđanja ishoda na novim, neviđenim podacima iznosi 52%. Iako postoji razlika u postocima točnosti predviđanja, ona nije tako jako velika. Iz grafa na Slici 20 možemo vidjeti i da je unakrsni gubitak entropije sveden na 0,5 što je prihvatljivo. Ovakvi su podaci ustvari i očekivani s obzirom na količinu prikupljenih podataka i malu razliku forme i rupica na listovima bolesnih palmi. Kruna zdrave i bolesne palme su gotovo identične u ranijim fazama zaraze palmine pipe, a razlike su uočljive samo u formi listova. Listovi zaraženih palmi su kao odrezani škarama na vrhovima, za razliku od zdravih listova kojima listovu zadržavaju obliju formu. Takve razlike će biti teško uočljive na malom broju palmi na kojima su se skupljali podaci.



Slika 19: Neuronska mreža modela 1 (Izrada autora, 2020)



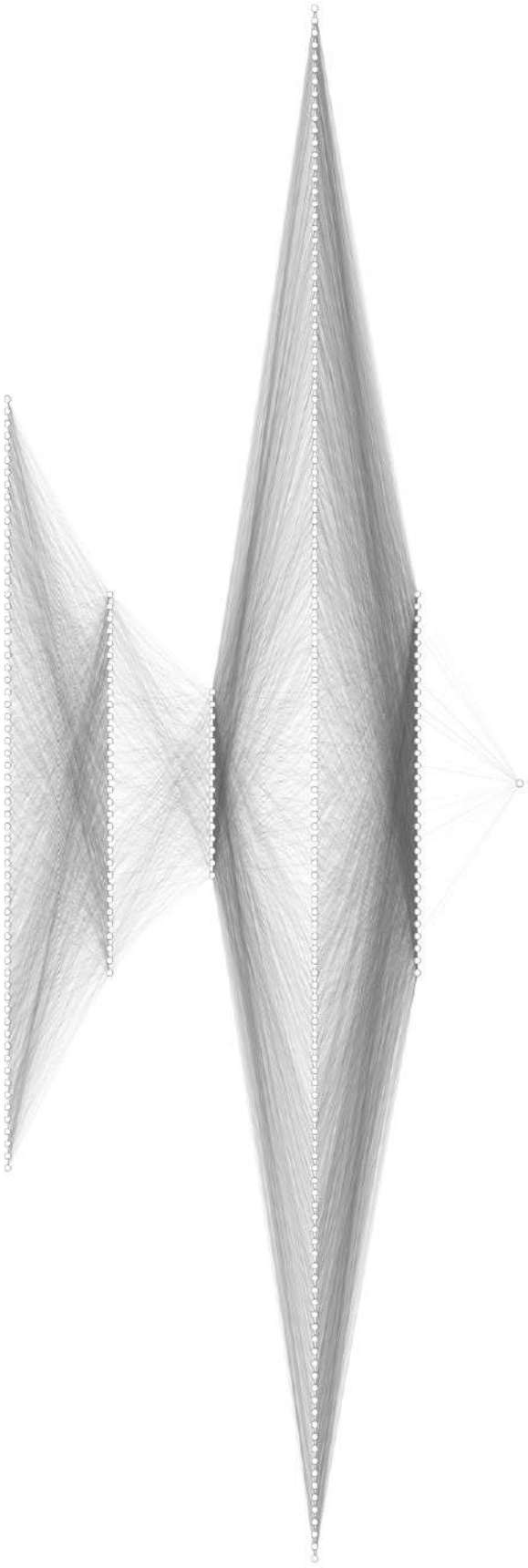
Slika 20: Unakrsni gubitak entropije i točnost klasifikacije modificiranog modela 1; plava - trenirani podaci, narančasta - testirani podaci (Izrada autora, 2020)

```
epochs = 32
batch_size = 10

def define_model():
    if K.image_data_format() == 'channels_first':
        input_shape = (3, img_width, img_height)
    else:
        input_shape = (img_width, img_height, 3)

    model = Sequential()
    model.add(Conv2D(16, (3, 3), input_shape=input_shape, activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(1, activation='relu'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

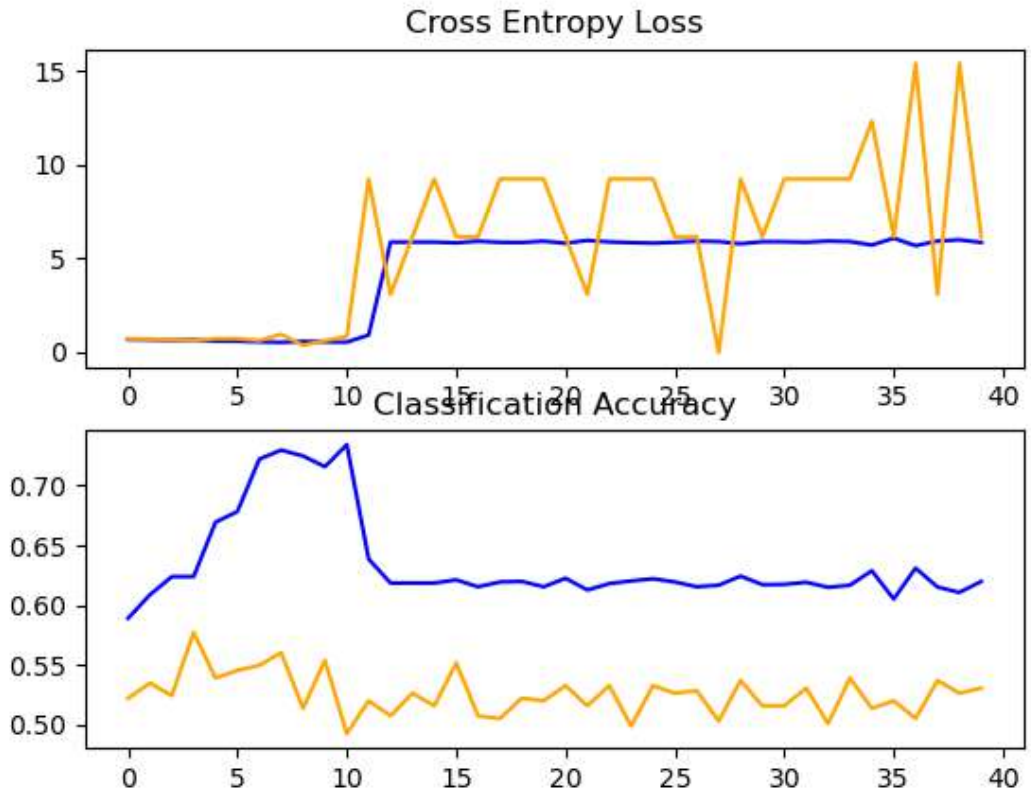
```
Epoch 1/32
224/224 [==============================] - 177s 791ms/step - loss: 0.7073 - accuracy
    : 0.5707 - val_loss: 0.7951 - val_accuracy: 0.5191
Epoch 2/32
224/224 [==============================] - 180s 803ms/step - loss: 0.6945 - accuracy
    : 0.6100 - val_loss: 0.5723 - val_accuracy: 0.5235
Epoch 3/32
224/224 [==============================] - 180s 804ms/step - loss: 0.6649 - accuracy
    : 0.6185 - val_loss: 0.7390 - val_accuracy: 0.4679
.
.
.
224/224 [==============================] - 169s 756ms/step - loss: 0.6614 - accuracy
    : 0.6257 - val_loss: 0.6245 - val_accuracy: 0.5321
Epoch 31/32
224/224 [==============================] - 169s 753ms/step - loss: 0.6656 - accuracy
    : 0.6176 - val_loss: 0.6250 - val_accuracy: 0.5278
Epoch 32/32
224/224 [==============================] - 169s 754ms/step - loss: 0.6653 - accuracy
    : 0.6179 - val_loss: 0.5323 - val_accuracy: 0.5171
> 52.092
```

Kako cijeli projekt ne bismo bazirali na samo jednom modelu, kreirali smo ih još nekoliko, a prikazati ćemo samo neke od njih. Sljedeći kreirani model je prikazan na Slici 21. Za razliku od modela 1 koji je u početku bio odličan primjer pretreniranosti, ovaj je model bio totalna suprotnost i jako dobro prikazuje problem nedovoljne naučenosti. Nedovoljna naučenost se događa kada je model puno jednostavniji od funkcije koja je generirala ulazne podatke, što znači da model ne uspijeva prepoznati sve ključne značajke, pa im se ne uspije ni prilagoditi.

```
Epoch 1/40
449/449 [==============================] - 192s 427ms/step - loss: 0.6847 - accuracy
    : 0.5890 - val_loss: 0.7227 - val_accuracy: 0.5221
Epoch 2/40
449/449 [==============================] - 185s 413ms/step - loss: 0.6678 - accuracy
    : 0.6087 - val_loss: 0.7131 - val_accuracy: 0.5349
Epoch 3/40
449/449 [==============================] - 180s 401ms/step - loss: 0.6338 - accuracy
    : 0.6238 - val_loss: 0.6547 - val_accuracy: 0.5243
Epoch 4/40
449/449 [==============================] - 190s 424ms/step - loss: 0.6616 - accuracy
    : 0.6238 - val_loss: 0.6415 - val_accuracy: 0.5772
.
.
.
Epoch 39/40
449/449 [==============================] - 177s 395ms/step - loss: 6.0119 - accuracy
    : 0.6106 - val_loss: 15.4249 - val_accuracy: 0.5264
Epoch 40/40
449/449 [==============================] - 1041s 2s/step - loss: 5.8539 - accuracy:
    0.6198 - val_loss: 6.1700 - val_accuracy: 0.5307

> 52.092
```



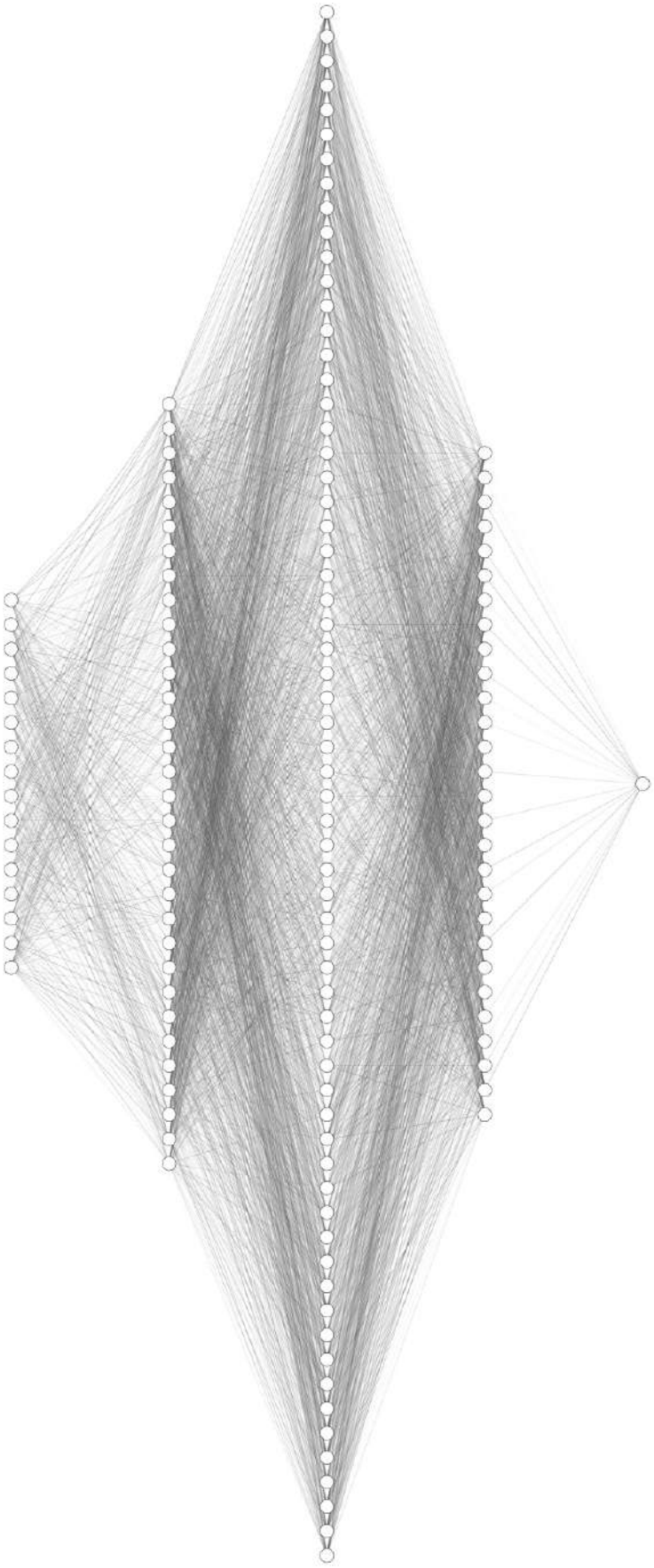
Slika 21: Neuronska mreža modela 2 (Izrada autora, 2020)



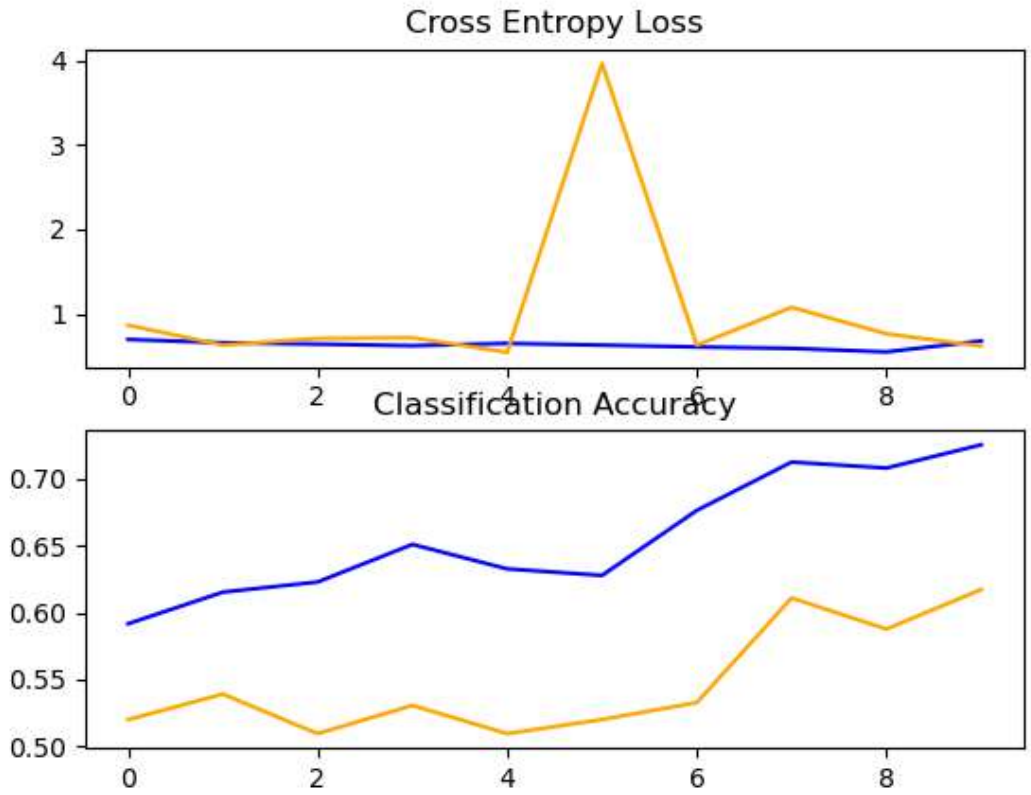
Slika 22: Unakrsni gubitak entropije i točnost klasifikacije modela 2; plava - trenirani podaci, narančasta - testirani podaci (Izrada autora, 2020)

Prema grafu sa Slike 24 možemo vidjeti kako je model ima ogromne entropijske gubitke. Iako je učenje krenulo obećavajuće, na epohi 10 se iznenadno smanjila točnost predviđanja klase, a entropijski gubitak je sve više i više rastao. Ovakvo ponašanje modela upućuje na nedovoljnu naučenost.

```
Epoch 1/10
449/449 [==============================] - 196s 435ms/step - loss: 0.6954 - accuracy
    : 0.5917 - val_loss: 0.8648 - val_accuracy: 0.5200
Epoch 2/10
449/449 [==============================] - 191s 426ms/step - loss: 0.6568 - accuracy
    : 0.6154 - val_loss: 0.6287 - val_accuracy: 0.5391
Epoch 3/10
449/449 [==============================] - 208s 463ms/step - loss: 0.6412 - accuracy
    : 0.6229 - val_loss: 0.7080 - val_accuracy: 0.5095
.
.
Epoch 9/10
449/449 [==============================] - 194s 432ms/step - loss: 0.5485 - accuracy
    : 0.7082 - val_loss: 0.7627 - val_accuracy: 0.5877
Epoch 10/10
449/449 [==============================] - 197s 438ms/step - loss: 0.6775 - accuracy
    : 0.7256 - val_loss: 0.6183 - val_accuracy: 0.6173
> 64.435
```



Slika 23: Neuronska mreža modela 3 (Izrada autora, 2020)



Slika 24: Unakrsni gubitak entropije i točnost klasifikacije modela 3; plava - trenirani podaci, narančasta - testirani podaci (Izrada autora, 2020)

Model sa Slike 23 se uspostavio kao najuspješniji model sa 64% točnosti predviđanja ishoda na novim, neviđenim zadacima. Rezultat treniranja modela se može vidjeti na Slici 24 i bloku kodu ispod grafa. Prema navedenim podacima vidimo da je entropijski gubitak podataka smanjen na 0.6 i da je točnost previđanja novih podataka 64%.

Prema narednom bloku koda, možemo vidjeti da se model ne razlikuje značajno od početnog modela. Jedina je razlika što sada normaliziramo podatke, a skriveni sloj od 128 čvorova, smo zamijenili slojem od 28 čvorova.

```
epochs = 10
batch_size = 5

def define_model():
    if K.image_data_format() == 'channels_first':
        input_shape = (3, img_width, img_height)
    else:
        input_shape = (img_width, img_height, 3)

    model = Sequential()
    model.add(Conv2D(16, (3, 3), input_shape=input_shape, activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(32, (3, 3), activation='relu'))
```

```
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    BatchNormalization()

    model.add(Flatten())
    model.add(Dense(28, activation='relu'))
    BatchNormalization()
    model.add(Dense(1, activation='relu'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    return model
```

Funkcija run_test_harness() nam služi za pokretanje testiranja kreiranog modela. Kao što je navedeno u poglavlju o transformaciji podataka, podatke za trening i testiranje dodatno povećavamo pomoću ImageDataGenerator() funkcije.

Kako bi generirali podatke za trening i testiranje koristimo funkciju flow_from_directory() koja uzima putanju do definiranih direktorija i generira skupine proširenih podataka. Funkciji proslijeđujemo putanju do direktorija gdje se nalaze podaci, zadajemo mu dimenzije na koje će se promijeniti sve pronađene slike iz direktorija, veličinu paketa podataka i možda najvažnije, postavljanje binary (pošto imamo samo dvije klase) kao vrstu polja oznaka koje se vraćaju.

Funkcija fit_generator nam omogućuje obučavanje modela na temelju podataka generiranih po paketima pomoću Python generatora. Kao argumente uzima paket podataka, broj koraka po pojedinoj epohi, broj epoha, validacijske podatke i broj koraka kod validacije. Nakon treniranja, model se pohranjuje i ispisuje se postotak točnosti predviđanja ishoda na novim neviđenim podacima.

```
def run_test_harness():
    # define model
    model = define_model()
    # create data generator
    # this is the augmentation configuration we will use for training
    train_datagen = ImageDataGenerator(
        rescale=1. / 255,
        shear_range=0.2,
        zoom_range=0.2,
        vertical_flip=True)

    # this is the augmentation configuration we will use for testing:
    test_datagen = ImageDataGenerator(
        rescale=1. / 255,
        vertical_flip=True)

    train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
```

```
        class_mode='binary')

    validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='binary')

    # fit model
    history = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)

    model.save('./model/final_model9.h5')
    # evaluate model
    _, acc = model.evaluate_generator(
        validation_generator, steps=len(validation_generator), verbose=0)
    print('> %.3f' % (acc * 100.0))
    # learning curves
    summarize_diagnostics(history)


# entry point, run the test harness
run_test_harness()
```

8.2.2. Crtanje grafova

Za prikupljanje statističkih podataka i crtanje grafova, koristili smo biblioteku matplotlib. Kreirali smo funkciju summarize_diagnostics koja kao parametar prima povijest treniranja modela. Podatke iz treninga bojamo plavom bojom, a podatke na kojima se model testira bojamo narančastom bojom. Funkiju summarize_diagnostics pozivamo na samom kraju programa, kada je model istreniran i validiran.

```
def summarize_diagnostics(history):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')
    # plot accuracy
    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue', label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange', label='test')
    # save plot to file
    filename = sys.argv[0].split('/')[-1]
    pyplot.savefig(filename + '_plotN.png')
    pyplot.close()
```

9. Proširenje sustava

Ukoliko se s povećanim brojem podataka ovaj sustav pokaže upotrebljivim u praktičnim situacijama, a ne samo u teorijskim simulacijama, mogao bi se proširiti. Sustav bi se proširio u smislu dodavanja nekih novih funkcionalnosti kao na primjer:

- poruke o stanju stanju palme (zdrava/zaražena) u stvarnom vremenu
- automatsko snimanje palmi bespilotnom letjelicom prema GPS koordinatama
- prepoznavanje faze zaraženosti palminom pipom
- prijedlozi preparata za tretiranje određenih faza zaraženosti palminom pipom
- evidencija tretiranja određene palme i praćenje palminog oporavka

Sustav za prepoznavanje bolesti palmine pipe bi se također mogao primijeniti i za prepoznavanje bolesti ostalih stabala ili možda čak određenih žitarica. Smatram da je ovo područje primjene strojnog učenja dosta neistraženo i da se tu krije velik broj interesantnih studija slučaja (eng. *case study*).

10. Zaključak

Strojno učenje je s razlogom jedna od najperspektivnijih grana informacijske tehnologije. Ovaj rad je pokušao metodama strojnog učenja prepoznati da li je neka palma zaražena nametnikom crvene palmine pipe. U jednu ruku je u tome i uspio. Najveću prepreku je stvarala mala količina podataka, zbog manjka setova podataka sa slikama palminih kruna slikanih iz zraka, većina je slika iz žablje perspektive. Pošto nije bilo druge opcije, krenulo se u sakupljanje podatka uz pomoć stručnih suradnika. S prikupljenim podacima, izmodelirano je nekoliko modela učenja. Od velikog broja kreiranih modela, neki su bili obećavajući dok su neki bili izrazito loši. Kreirano je par modela koji su gotovo savršeno pokazali probleme pretreniranosti i premale naučenosti.

Rezultati treniranja dobrih modela varirali su između 53% i 64%. Takvi su postoci točnosti previđanja realni, pošto su razlike u zdravim i zaraženim palmama izrazito male. Smatram da bi se sa povećanjem broja ulaznih podataka, tj. sa većim brojem zdravih i zaraženih palmi, ovaj postotak povećao i da bi se ovaj sustav mogao koristi u praktičnim situacijama, a ne samo kao teorijska simulacija. Proces prikupljanja i transformacije podataka, isto kao i proces kreiranja, treniranja i testiranja modela učenja je detaljno opisan uz prikaz blokova koda. Sve je tako detaljno opisano kako bi čitatelj dobro razumio procese, ali i opseg ovog projekta.Korištenje ovog sustava bi agronomima uvelike pomoglo kod detekcije prisutnosti štetnika u palmi i samim time spasilo palmu od potencijalnoga umiranja.

Iako je ovaj projekt ostavio dosta prostora za poboljšanje i nadogradnju, smatram da je dokazao izvedivost ovog koncepta. Umjetna inteligencija je u svojoj životnoj fazi u kojoj se potiho ugnježđuje u svakodnevni život ljudi do te mjere da za desetak godina, život bez umjetne inteligencije biti će nepojmljiv čovjeku.

Popis literature

[1] „Keras”, 2020. adresa: https://keras.io/ (pogledano 1. 9. 2020).

[2] „NumPy”, 2020. adresa: https://numpy.org/ (pogledano 1. 9. 2020).

[3] „OpenCV”, 2020. adresa: https://opencv.org/ (pogledano 1. 9. 2020).

[4] „Matplotlib”, 2020. adresa: https://matplotlib.org/ (pogledano 3. 9. 2020).

[5] „os — Miscellaneous operating system interfaces”, 2020. adresa: https://docs.python.org/3/library/os.html (pogledano 3. 9. 2020).

[6] „Phantom 3 Standard - Drone for Beginners - DJI”, 2020. adresa: https://www.dji.com/hr/phantom-3-standard (pogledano 3. 9. 2020).

[7] M. M. Tatjana, *NAJVAŽNIJI ŠTETNICI PALMI crvena palmina pipa i palmin drvotoč*. HCPHS - Zavod za zaštitu bilja, 2016.

[8] „Rhynchophorus ferrugineus (red palm weevil)”, 2020. adresa: https://www.cabi.org/isc/datasheet/47472 (pogledano 23. 8. 2020).

[9] S. Alex i V. S.V.N., *Introduction to Machine Learning*. Cambridge University Press, 2008.

[10] N. Nils J., *INTRODUCTION TO MACHINE LEARNING*. Stanford University, 2005.

[11] S.-S. Shai i B.-D. Shai, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[12] G. Richard, *Psychology: The Science of Mind and Behaviour 6th Edition*. Hodder education, 2010.

[13] J. Gareth, W. Daniela, H. Trevor i T. Robert, *An Introduction to Statistical Learning*. Springer, 2017.

[14] B. Jason, „4 Types of Classification Tasks in Machine Learning”, 2020. adresa: https://machinelearningmastery.com/types-of-classification-in-machine-learning/ (pogledano 1. 9. 2020).

[15] N. Michael, *Neural Networks and Deep Learning*. Determination Press, 2018.

[16] „What is a camera’s frame rate, or fps?”, 2020. adresa: https://www.videosurveillance.com/tech/frame-rate.asp (pogledano 3. 9. 2020).

Popis slika