SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij računarstva

RAČUNARSTVO USLUGA I ANALIZA PODATAKA

Prepoznavanje rukom pisanih brojeva (OCR)

Seminarski rad

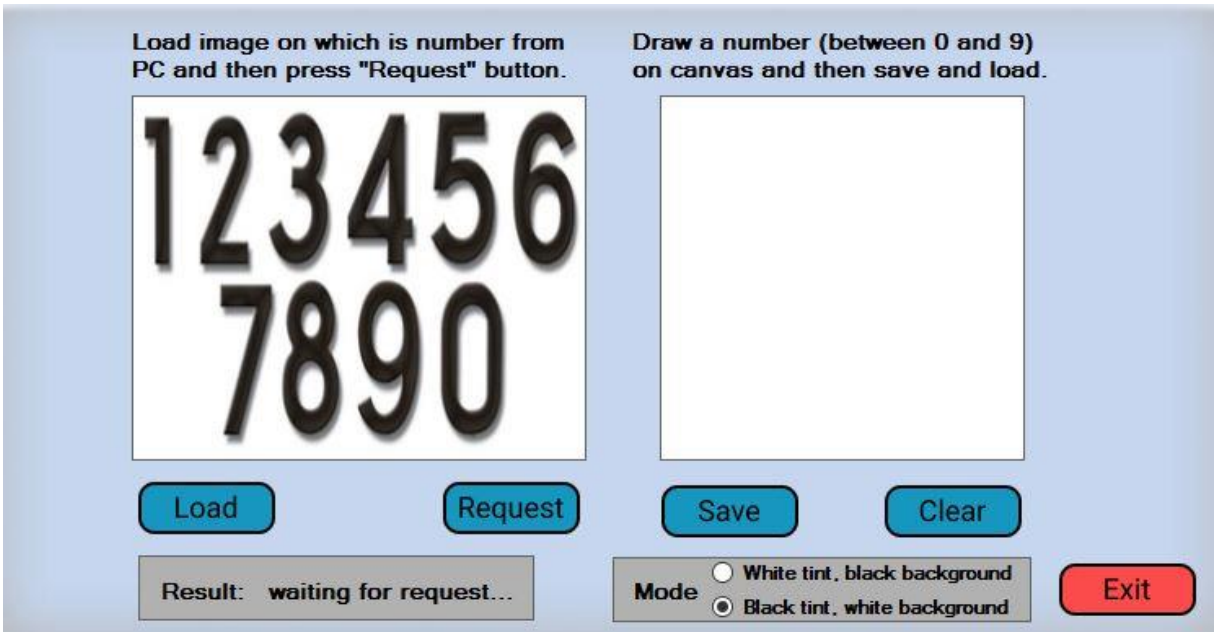
Ivan Drulak

Osijek, 2018.

Sadržaj

1. UVOD

Kao glavni cilj ovog projektnog zadatka je omogućiti prepoznavanje rukom pisanih brojeva sa slike. Pritom je potrebno proučiti moguće probleme i način na koji se ti problemi mogu riješiti. Znamo da brojevi nisu uvijek jednako napisani i na prvu jasno prepoznatljivi. U ovom projektu promatrane su vrijednosti piksela sa slike na temelju kojih se donosi odluka koji je broj prikazan na slici. Jedan od bitnih faktora je dobro poznavanje skupa podataka nad kojim će se primjenjivati strojno učenje. Kako se radi o prepoznavanju na temelju unaprijed poznatih značajki koristi se klasifikacija. Znamo da klasifikaciju koristimo kada želimo ulazne podatke razvrstati u pojedine skupine, grupe, a pritom su nam poznati ulazni podaci, u kojem obliku bi trebali biti. Kao proširenje zadatku postoji mogućnost korisnika da nacrta neki od brojeva od 0 do 9 i da pritom spremi taj broj kao sliku i zatim ispita hoće li broj biti točno određen. U ovom projektu nije moguće prepoznavanje više brojeva ukoliko se nalaze na jednoj slici nego samo jednog broja.



Sl. 1.1. Prikaz korisničkog sučelja aplikacije

1.1. SKUP PODATAKA

Korišteni skup podataka za ovaj projekt sadrži 785 značajki. Razlog tako velikog broja značajki je taj što su spremljene slike veličine 28x28 piksela što čini 784 značajki koje se odnose na vrijednosti pojedinog piksela na slici i još jedna značajka je sami broj koji je stvarno nalazi na slici. Vrijednosti piksela su u rasponu od 0 do 255, pri čemu vrijednost u ovom skupu podatka 255 predstavlja najtamniju vrijednost piksela (crna boja), a 0 najsvjetliju (bijela boja). Pošto svaki piksel je bitan nije bilo moguće raditi normalizaciju podataka niti smanjivati značajke. Svaka značajka ima naziv „pikselx“ pri čemu x predstavlja broj od 0 do 783 kao redni broj piksela sa slike. Ovaj naziv je bitan kako bi se prilikom slanja zahtjeva servisu predala vrijednost pojedinog piksela sa potrebnim nazivom koji zahtjeva web servis u suprotnom servis neće moći izvršiti strojno učenje i vratiti rezultat koji predstavlja predviđeni broj sa slike. Ovaj skup podataka preuzet je s Kaggle[1]-ove stranice.

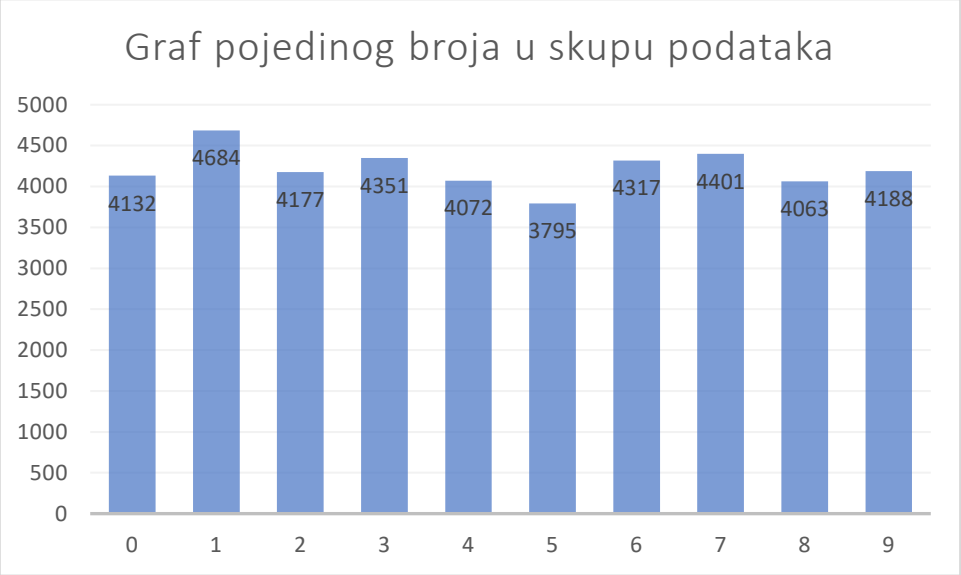
Broj	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10
1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0

Sl. 1.1.1. Prikaz prvih 10 vrijednost piksela i broj koji čine ti pikseli

Na slici iznad (Sl. 1.1.1.) prikazano je samo 10 značajki iz razloga što nije moguće prikazati slikom svih 784 značajki. Možemo vidjeti da svaki broj za prvih 10 piksela ima vrijednost 0. Razlog je tome što niti jedan broj nije na početku slike nego se nalazi u sredini. Sa slike to nije moguće vidjeti, ali prvih 50 i zadnjih 50 piksela uvijek imaju vrijednost 0 pošto se ne nalazi ništa osim bijele boje. Kako je već ranije spomenuto niti jedna od značajki se ne smije izbaciti

[1] Izvor skupa podataka: https://www.kaggle.com/bdaubney/numbers/data

jer bi uveliko utjecalo na kvalitetu prilikom korištenja strojnog učenja nad ovim skupom podataka jer je svaki piksel bitan.



Sl. 1.1.2. Prikaz ukupnog broja pojedinog broja u skupu podataka

Na grafu sada jasno možemo vidjeti koliko ima primjerka značajki pojedinog broja od 0 do 9 nad cijelim skupom podatka. Za svaki pojedinačni broj postoji više od 4 000 primjeraka osim za broj 5 koji ima nešto manje od 4 000 primjeraka. Možemo reći da ovaj skup podataka ima poprilično ravnopravnu raspodjelu podataka za treniranje.

Primjeri slike brojeva iz skupa podataka:

Sl. 1.1.3. Prikaz rekonstruiranih rukom pisanih brojeva iz skupa podataka

1.2. MODELI STROJNOG UČENJA

Kako bi dobili što bolje rezultate klasificiranja brojeva bilo je potrebno isprobati različite algoritme strojnoga učenja i time isprobati različite modele ne bi li time došli do što boljih rezultata i modela koji je najbolji za ovaj tip klasificiranja. Prije prosljeđivanja podataka algoritmima bilo je potrebno razdvojiti podatke na dio nad kojima će se izvršiti treniranje te na dio koji predstavlja testiranje, točnosti predviđanja rezultata. Za svaki od algoritama korišten je odnos 70% za treniranje i 30% za testiranje (predviđanje) rezultata. Ulazni podaci su spremljeni u .csv format pri čemu postoji header koji se sastoji od dva dijela: stvarni broj i „pixelx“ kako je već ranije spomenuto. Svaki od algoritama je zatim trenirao nad istim podacima pri čemu su se radile promijene na njihovim parametrima kako bi se dobili što bolji rezultati. Korišteni algoritmi za strojno učenje u ovom projektu su: Multiclass Decision Forest, Support Vector Machine, Multiclass Decision Jungle i Multiclass Neural Network. Sada slijedi kratko objašenje svakog algoritma i rezultati njihovih predviđanja rukom pisanih brojeva.

1.2.1. Multiclass Decision Forest

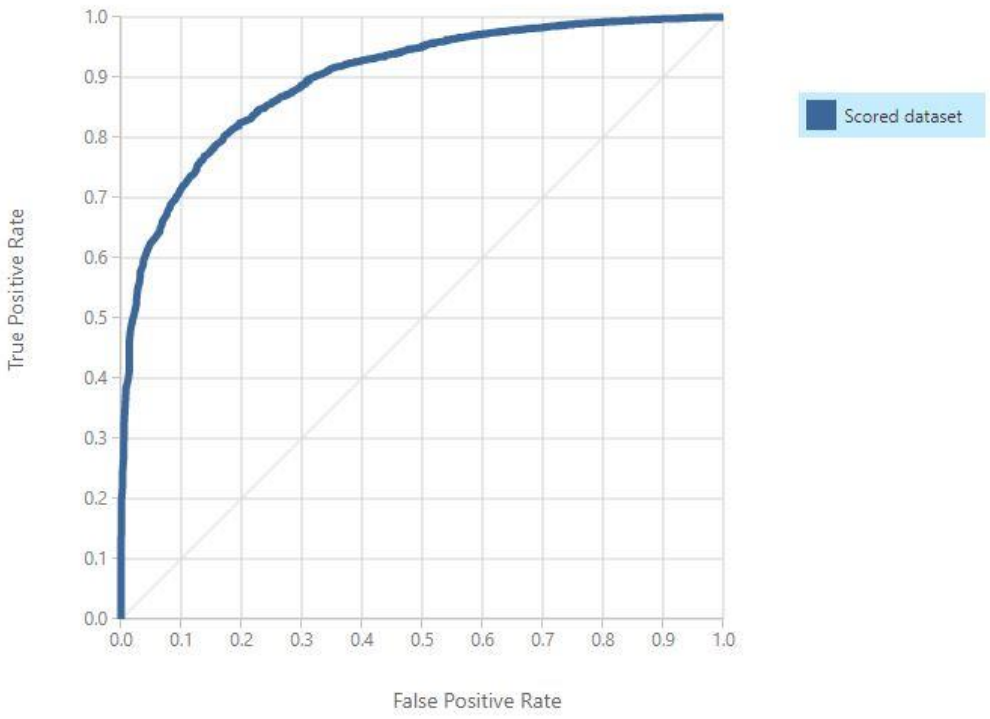
Multiclass Decision Forest sastoji se od više stabla koja odlučuju, daju rezultat što je rješenje. Pošto se ovdje radi o brojevima i vrijednostima piksela sa slike svaki čvor stabla će zapravo biti jedan piksel. Ovaj algoritam za svako stablo radi grananje na čvorove sve dok se ne dobije jasan rezultat (jedan od brojeva od 0 do 9). Postavljeno je za parametar ukupno 256 pojedinačnih stabla pri čemu svako stablo kao izlaz daje predviđeni broj kao rješenje i onaj broj koji je najviše puta predviđen predstavlja konačno rješenje. Razlog zašto je korišteno baš 256 stabla, a ne manje ili više je to što je prilikom korištenja većeg broja bilo je potrebno puno vremena da se istrenira algoritam, a nije postigao znatno bolje rezultate te je prilikom korištenja manjeg broja došlo do znatno slabije točnost predviđanja. Na idućoj stranici nalazi se slika sa matricom grešaka na temelju koje možemo vidjeti koliko su dobro predviđeni brojevi i u kojem postotku su pojedini brojevi krivo određeni.

Sl. 1.2.1. Prikaz matrice pogrešaka za model s Multiclass Decision Forest

Iz slike iznad možemo vidjeti da se na glavnoj dijagonali nalaze plavom bojom označeni postoci koji označavaju koliko su uspješno određeni pojedinačni brojevi. Na y osi nalaze se točni brojevi sa slike, a na x osi predviđeni brojevi. Za broj 0 vidimo da je ukupan postotak 98.1% što znači da je za svega 1.9% krivo predviđeno da je neki drugi broj. U prvom redu algoritam je nulu zamijenio sa brojevima: 2, 5, 6, 8 i 9 pri čemu je najviše puta zamijenio s brojem 8 (1%). Isto tako slijedno se dalje mogu vidjeti kriva predviđanja za ostalih osam brojeva. Ovakav algoritam je poprilično zadovoljavajuće predvidio brojeve pri čemu mu ukupna preciznost iznosi 96.59%.

1.2.2. Support Vector Machine

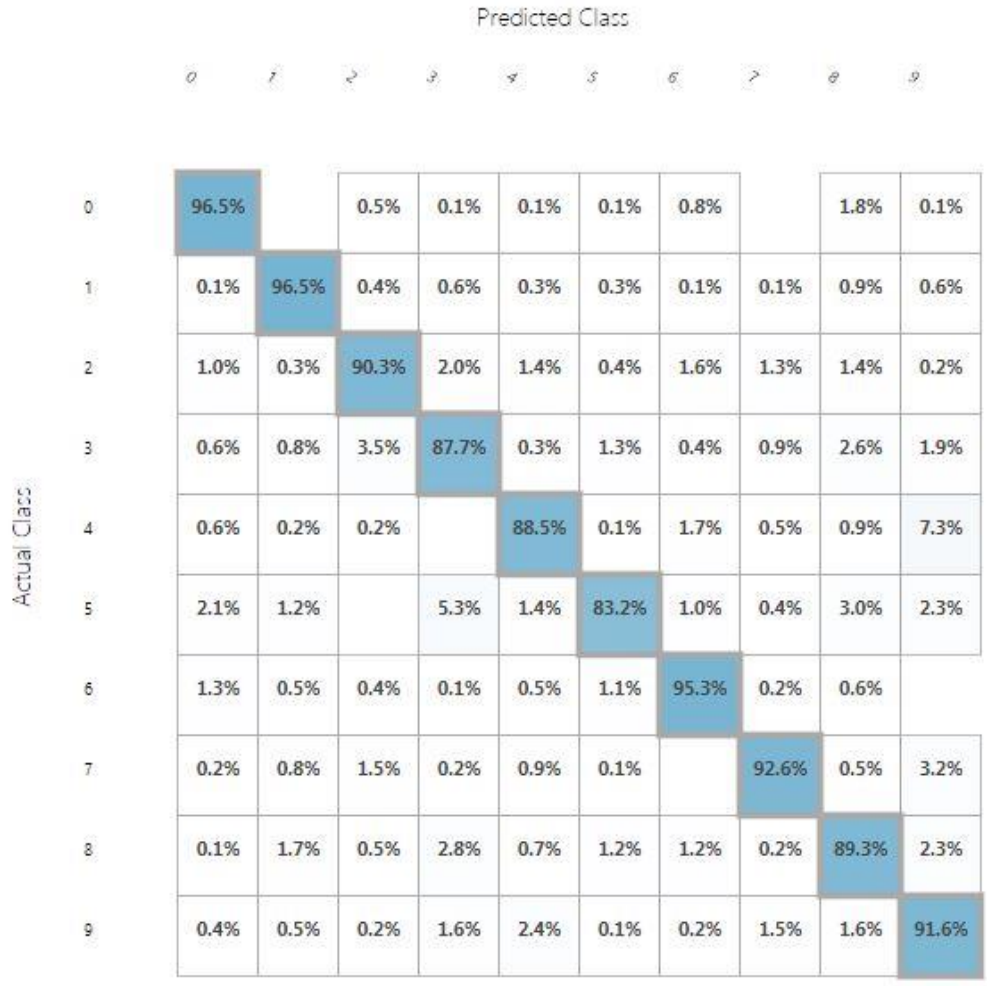
Začetnik ovog algoritma je Vladimir N. Vapnik koji je razvio matematički model koji na temelju vektora može omogućiti klasificiranje podataka. SVM se temelji na pomočnim vektorima i hiper reavnini koja odjeljuje podatke u grupe, klase. Hiper ravnina u 2D prostoru predstavlja pravac, 3D ravninu, a u N-dimenzijonalnom prostoru hiper ravninu koja je jednog reda manja. Glavni parametar je λ (Lambda) na temelju koje se definira generalizacija klasifikacije. Jednostavnije rečeno, ukoliko je vrijednost tog parametara veća doći će do bolje generalizacije podataka, a ukoliko je dovoljno mala doći će do lokalnog određivanja podataka točnije do takozvanog overfitting-a. Iterativnim postupkom pokazalo se da pri vrijednosti $\lambda = 0.01$ postiže najveća točnost algoritma od 92.5%.



Sl. 1.2.2. Prikaz grafa za SVM pri čemu se prikazuje odnos dobih i loših predviđanja

1.2.3. Multiclass Decision Jungle

Multiclass Decision Jungle predstavlja proširenje i nadogradnju za Multiclass Decision Forest pri ćemu se koristi DAG (usmjereni aciklički graf). Takvi grafovi omogućuju da se jedan čvor može imati više roditelja i time poboljšati i smanjiti ukupni broj čvorova. Time se postiže smanjenje potrebnih resursa računala pri čemu je jedan od najkritičnijih faktora memorija. Kada se radi s velikom količinom podataka (Big Data) tada Decision Forest algoritam zahtijeva previše memorije i dolazi do zagušenja te smanjenja učinkovitosti pri izvođenju. Zbog toga je nastalo ovo proširenje korištenjem DAG-ova. Za ovaj je algoritam broj DAG-ova postavljeno na 256, pri čemu je dobivena točnost od 91.27% što je nešto više od SVM-a. Ispod se nalazi slika matrice pogrešaka kao rezultat točnosti korištenjem ovog algoritma.



Sl. 1.2.3. Prikaz matrice pogrešaka za model s Multiclass Decision Jungle algoritmom

1.2.4. Multiclass Neural Network

Multiclass Neural Network neuronska je mreža koja omogućuje rad s više klasa, grupa podataka te se sastoji od više slojeva. Može se reći da je model simetričan jer s jedne strane postoji sloj koji predstavlja ulaz, a s druge strane sloj koji predstavlja izlaz. Između se nalaze višestruki skriveni slojevi koji međusobno razmjenjuju podatke i prosljeđuju jedni drugima (od prethodnog do sljedećeg). Svaki sloj ima čvorove koji su međusobno povezani težinama. Također, svaki čvor izračunava vrijednost težine kao brojčane vrijednosti i prosljeđuje je idućem čvoru, odnosno sloju. U ovom projektu početni sloj, točnije ulaz predstavlja 784 piksela, odnosno čvorova (neurona) u kojem su sadržane vrijednosti svakog piksela (slika je veličine 28x28 piksela). Bitno je naglasiti da su vrijednosti unutar čvora u rasponu od 0 do 1. Ukoliko je vrijednost 0 tada je čvor „ne aktivan“, ali ukoliko je vrijednost veća tada postaje „aktivan“ i utječe na krajnji rezultat. Zatim slijede skriveni slojevi pri čemu određena kombinacija aktivnih čvorova u jednom skrivenom sloju prenosi svoju vrijednost u drugi sloj i također aktivira određene čvorove, pri tome stvarajući određeni uzorak za svaki ulaz. Svaki takav uzorak rezultira rješenjem na izlazu, pri čemu se predviđa koji je broj sa slike na ulazu. Naravno to nije tako jednostavno i zato se koriste težine (bilo koja brojčana vrijednost koju mreža sama definira na temelju povratne veze o uspješnosti klasifikacije prilikom treniranja) i bias (broj koji definira za koju će vrijednost sume čvor biti aktivan), koji zajedno omogućuju bolje definiranje kada će pojedini čvor biti aktivan pri čemu će u idealnom slučaju rezultirati uvijek točnim rješenjem. Ukupna točnost ovog modela je 97.82% što je najviše od svih prethodnih. Na stranici koja slijedi nalazi se slika koja prikazuje matricu pogrešaka. Jasno se može vidjeti da su vrijednosti na glavnoj dijagonali u najvećem postotku do sada.

Sl. 1.2.4. Prikaz matrice pogrešaka za model s Multiclass Neural Network algoritmom

Na temelju svih korištenih algoritama i modela vidimo da je neuronska mreža ostvarila najbolje rezultate. Jedini je model koji je dostigao točnost preko 97%, što nikako nije beznačajno. Kao najlošiji algoritam pokazao se Multiclass Decision Jungle čija je točnost iznosila 91.27%, zatim SVM s točnošću od 92.5% i kao posljednji najbliži neuoronskim mrežama je Multiclass Decision Forest s točnošću od 96.59%. Jedan od mogućih razloga zašto je Jungle algoritam postigao najlošije rezultate je zbog toga što kod vrijednosti piksela slike treba iskoristiti i proći kroz sve piksele što vjerojatno nije slučaj pošto se koriste ciklički grafovi koji smanjuju grananje i time se ne koriste svi pikseli. S druge strane, kod Forest algoritma nema takve optimizacije i time se koriste svi podaci (pikseli) slike i postižu se u konačnici bolji rezultati. Iznenađujuće slabe rezultate postigao je SVM, koji je prvobitno isključivo i testiran za klasifikaciju brojeva te kao takav postao zapažen u svijetu strojnog učenja. Jedan od razloga je slaba mogućnost postavljanja parametara za ovaj algoritam. Mogu se samo postaviti parametri za broj iteracija, vrijednost λ, normalizacija značajki što nisu svi parametri koji karakteriziraju ovaj algoritam. Jedan od važnih parametara koji se ovdje ne mogu namjestiti je C (penalty error) koji omogućuje koliko se dopušta odstupanje podataka kao točna klasifikacija.

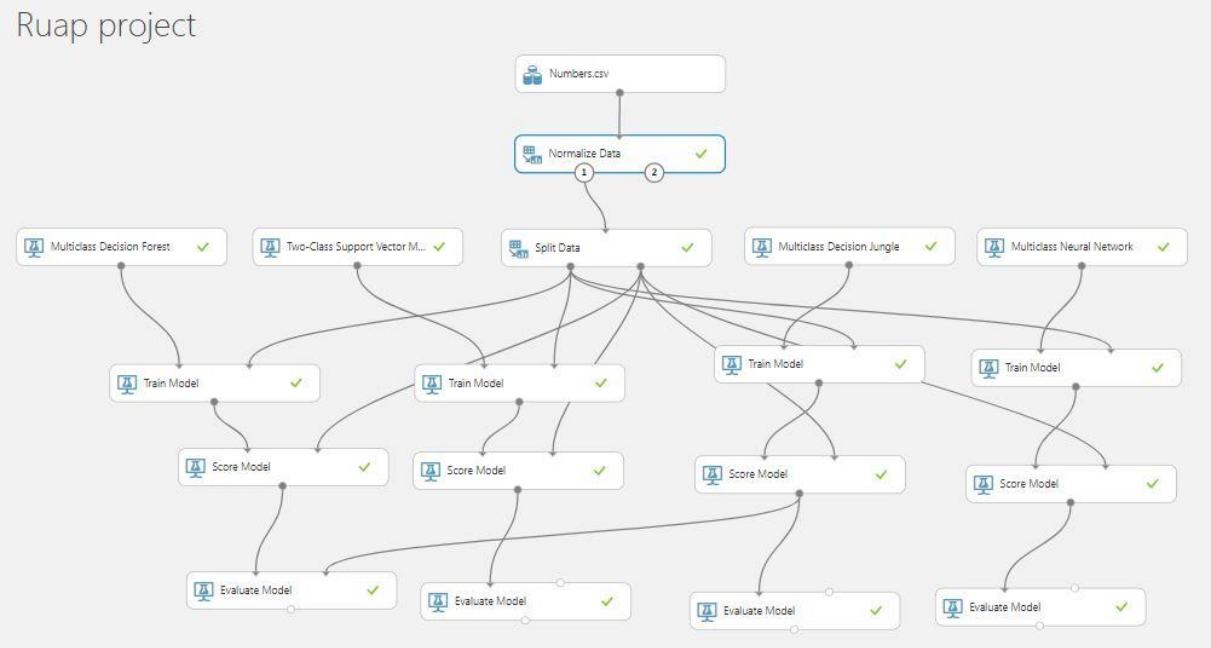
2. OPIS PROGRAMSKOG RJEŠENJA

Za programsko rješenje uzet je model strojnoga učenja pri čemu je korištena neuronska mreža koja se pokazala kao najbolja od svih ostalih modela koji su isprobani i testirani. Model je napravljen u Microsoft Azure Machine Learning Studio-u. Ovaj model služi za prepoznavanje rukom pisanih brojeva sa slike. Glavne značajke brojeva su vrijednosti piksela od kojih je sastavljena svaka slika i ukupno ima 784 značajki. Na kraju je cijeli model implementiran kao web servis s kojim se može komunicirati putem API ključa za autorizaciju i određenim zahtjevima za slanje i primanje odgovora.

Aplikacija koja je povezana sa servisom i omogućuje razmjenu podataka napravljena je unutar C# Windows Forms Application. Ova aplikacija predstavlja sučelje između servisa i korisnika. Korištena su dva dodatna paketa, prvi je EmguCV-a koji omogućuje rad sa slikama, a zapravo predstavlja C# implementaciju Python biblioteke OpenCV. Kao drugi paket tu je ASP.NET Web API koji omogućuje rukovanje Http zahtjevima i korištenje Azure web servisa.

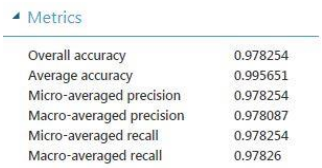
2.1. MODEL STROJNOG UČENJA

Kako je već ranije spomenuto kao rješenje uzet je model s neuronskom mrežom koja se pokazala kao najbolje rješenje. Naravno, isprobani su i drugi algoritmi, odnosno modeli, kako je navedeno u prethodnom podpoglavlju 1.2. U nastavku slijedi slika koja prikazuje sve modele.

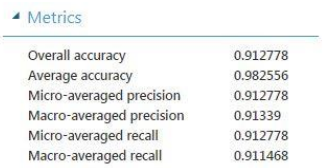


Sl. 2.1. Modeli strojnoga učenja

Slika 2.1. prikazuje sve modele koji su isprobani u ovom projektu. Promatrajući sliku od gore prema dolje možemo vidjeti da se na početku nalazi skup podataka pod nazivom Numbers.csv. On predstavlja početak modela i izvor podataka na temelju kojih se treniraju modeli i provjerava njihova točnost, preciznost, matrica pogrešaka itd. Zatim slijedi normalizacija podataka pri čemu je odabran način MinMax koji sve vrijednosti piksela od 0 do 255 stavlja u raspon od 0 do 1. Samom normalizacijom nije postignuto neko značajno poboljšanje u točnosti algoritama, ali je samo treniranje modela bilo nešto brže. Zatim slijedi razdvajanje podataka (Split Data) po retcima pri čemu je 70% namijenjeno za treniranje, a 30% za testiranje. Nakong toga se tako distribuirani podaci dalje prosljeđuju svakom modelu za treniranje (Train Model) koji je povezan sa odgovarajućim algoritmom strojnog učenja. Unutar Train Model-a potrebno je odabrati značajku koja predstavlja onu koju je potrebno klasificirati, a to je u ovom slučaju broj. Takav istrenirani model zatim prosljeđuje rezultate u Score Model unutar kojeg se inače mogu vidjeti sve značajke korištenog skupa podataka, vjerojatnosti pojavljivanja određenih značajki i najbitnije rezultat klasifikacije, strojnog učenja općenito (Scored Labels). Nažalost pošto ovaj skup podataka ima jako puno značajki (784) nije ih moguće vidjeti i prikazati grafom kao rješenja. Razlog tome je što Score Model može prikazati samo 100 različitih značajki. Na temelju odabranih značajki mogu se prikazivati grafovi (histogrami) na kojima se mogu vidjeti frekvencije pojavljivanja označenih značajki te se također mogu uspoređivati različite značajke. Kao posljednji aktivni element nalazi se evaluacijski model (Evaluate Model) koji prikazuje metričke podatke o točnosti i preciznosti modela i matrice pogrešaka koja pokazuje u postotku koliko je dobro klasificiran pojedini broj sa slike. Kod evaluacijskog modela mogu se prikazati rezultati dva modela i time odmah usporediti koji je model bolji i u čemu točno. U nastavku slijedi primjer prikaza rezultata modela pri čemu su korišteni algoritmi: Multiclass Decision Jungle i Multiclass Neural Network.



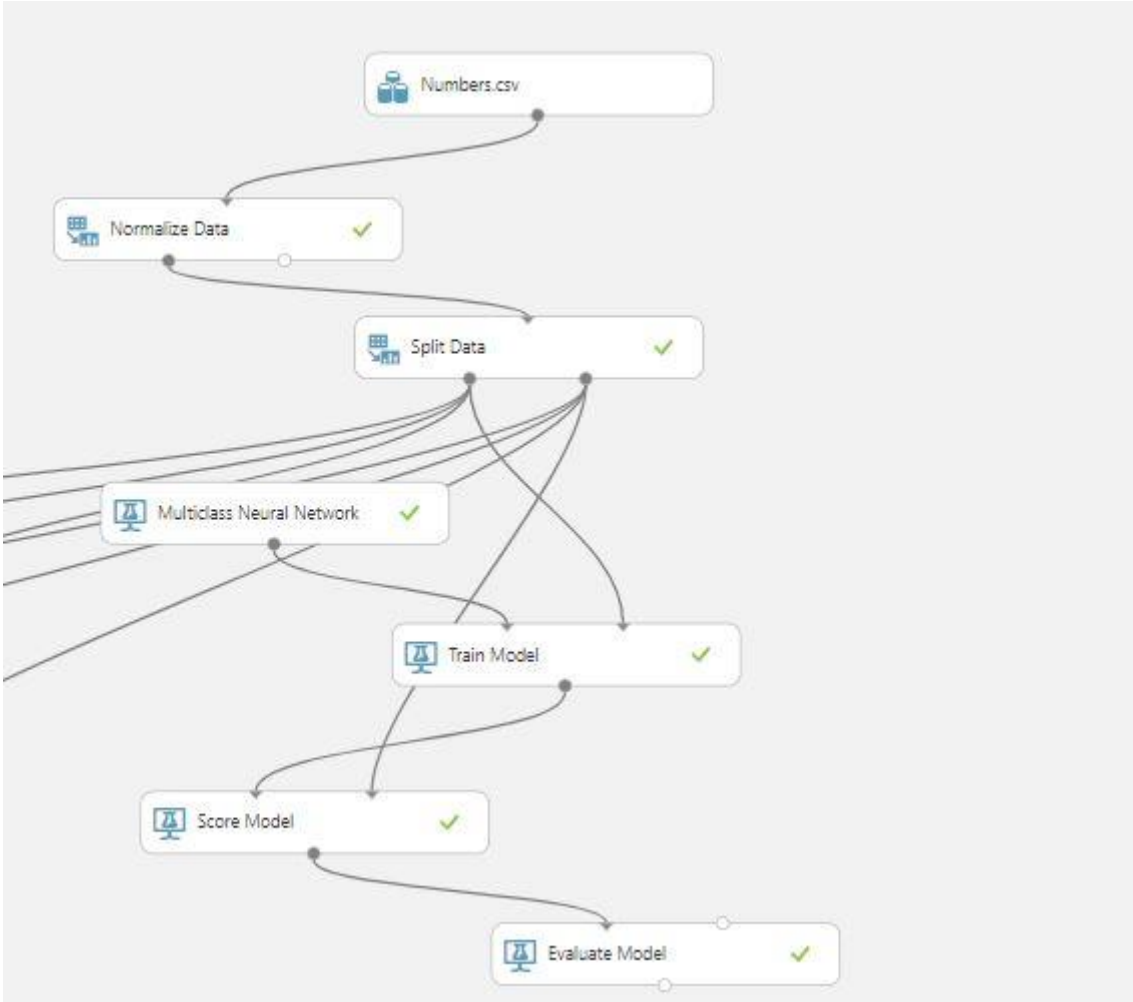
Metrics	
Overall accuracy	0.978254
Average accuracy	0.995651
Micro-averaged precision	0.978254
Macro-averaged precision	0.978087
Micro-averaged recall	0.978254
Macro-averaged recall	0.97826



Metrics	
Overall accuracy	0.912778
Average accuracy	0.982556
Micro-averaged precision	0.912778
Macro-averaged precision	0.91339
Micro-averaged recall	0.912778
Macro-averaged recall	0.911468

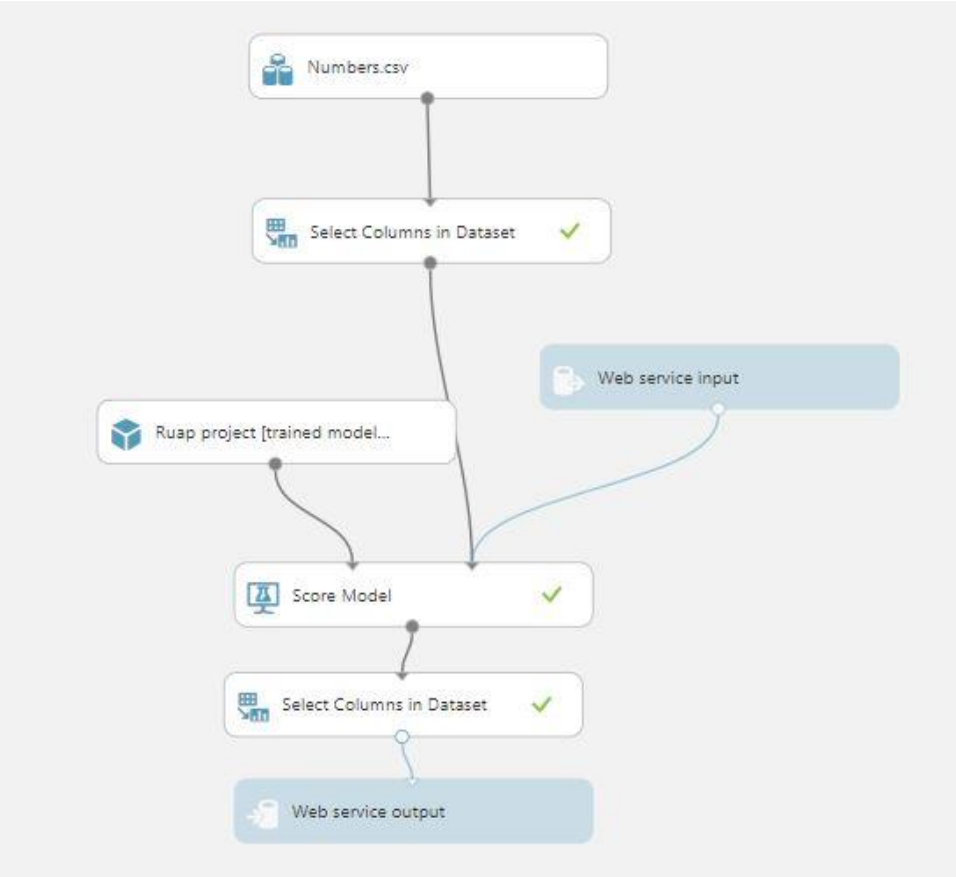
Sl. 2.2. Prikaz rezultata – Evaluate Model-a za Decision Jungle i Neural Network

Na temelju slike s prethodne stranice vidimo da je po svim metričkim parametrima rezultat modela s neuronskom mrežom bolji kao i kod grafičkog prikaza matrice pogrešaka. Kod matrice pogrešaka za neuronsku mrežu broj 0 nikada nije bio klasificiran kao broj 1, broj 1 nikada nije bio klasificiran kao broj 0, 5 i 9, i tako redom možemo vidjeti za ostale brojeve. Na kraju vidimo da je puno manje pogrešnih klasificiranja nego kod algoritma Decision Jungle, i naravno, manji je postotak pogrešnog klasificiranja. Ispod se nalazi uvećana slika modela koji je korišten za strojno učenje i implementiran kao web servis.



Sl. 2.3. Prikaz modela s neuronskom mrežom

Nakon što je odabran model on je implementiran kao web servis na Azure-u te koristi HTTP protokol kao način komuniciranja putem interneta. Time je web servis stvorio sam pristupni ključ za autorizaciju korisnika, dodatni drugi ključ koji se najčešće koristi prilikom izmjene i nadogradnje servisa i kada se žele promijeniti pristupni ključevi, a da je pritom i dalje omogućen pristup servisu. Najbitniji za komuniciranje sa servisom je API ključ za slanje podataka i primanje odgovora. Servis kreira i mogućnost korištenja batch zahtjeva koji se koriste kada se želi poslati veći broj API zahtjeva prema servisu pomoću jednog HTTP zahtjeva. Kao pomoć generiran je i kod za komuniciranje sa servisom u: C#, Python i R jeziku. Za ovaj projekt koristi se C# programski jezik za komuniciranje sa servisom.



Sl. 2.4. Prikaz web servisa napravljenog iz modela sa neuronskom mrežom

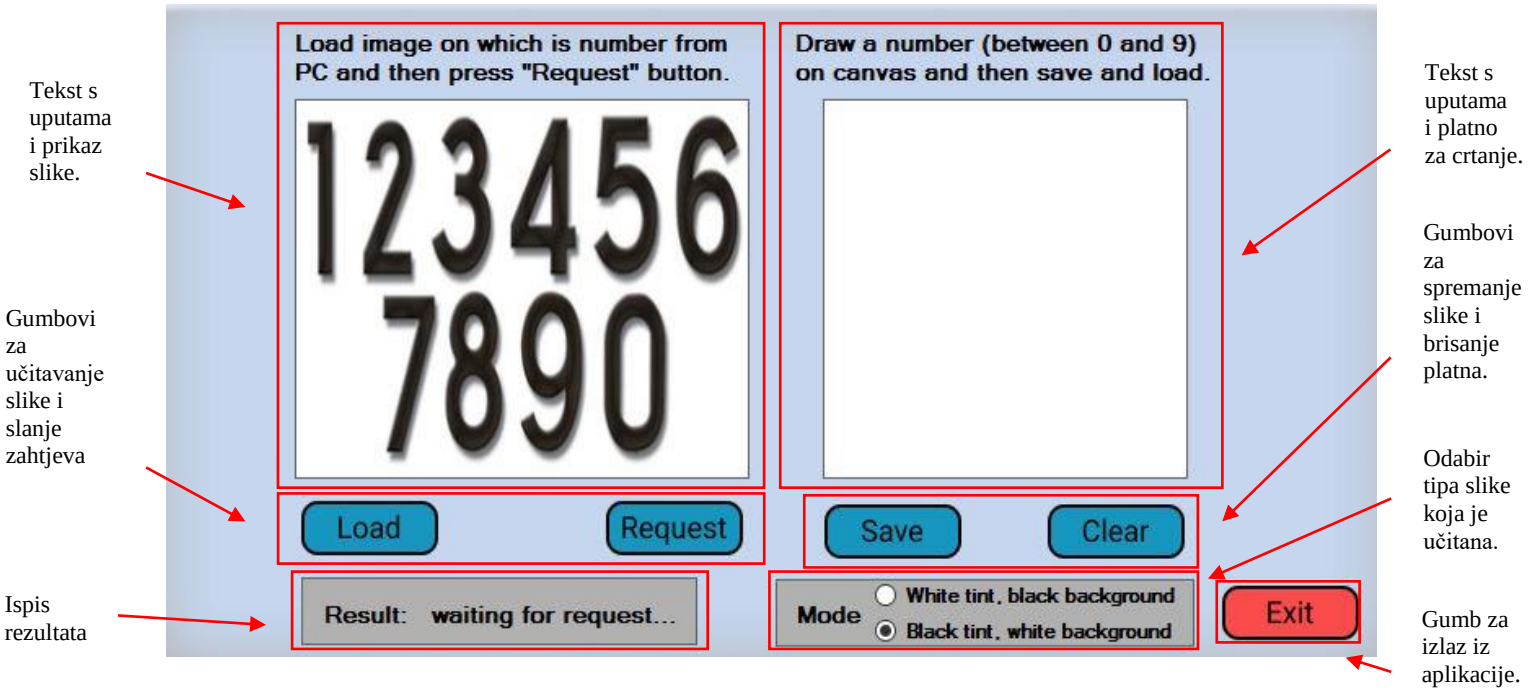
Na prikazu iznad možemo vidjeti da se na početku nalazi skup podataka pri čemu se odabiru stupci iz skupa podataka, a to su značajke (svi pikseli). Dalje je sve povezano sa Score Model-om koji dobiva rezultate iz odabranog istreniranog modela (neuronska mreža) te je također povezan s ulazom koji očekuje unos 784 piksela. Prije samog slanja rezultata servisa na izlaz prema željenom odredištu (u ovom slučaju prema C# Forms aplikaciji, korisniku) postoji aktivni element (Select Columns in Dataset) unutar kojeg se mogu odabrati stupci koji se žele poslati na izlaz. Ovaj element je korišten pošto prilikom testiranja i rada se servisom osim rezultata (Scored Labels, broj koji je predviđen) postavljeno je da servis šalje i postotak vjerojatnosti pojedinog broja (od 0 do 9) za koji algoritam predviđa da je na slici.

2.2. API KORIŠTENJE

Kako bi mogli koristiti definirani model za klasificiranje broja sa slike potrebno je moći pristupiti web servisu. Za pristup je potrebno imati primarni ključ (API ključ) i znati API za slanje zahtjeva i primanje odgovora. Request-Response API: https://ussouthcentral.services.azureml.net/workspaces/70f52eba71314efeb7ddf06882d4ac2c/services/b5c32c7f2acf4c939f327063a2c4533b/execute?api-version=2.0&format=swagger Unutar zaglavlja za Request-Response metodu potrebno je koristiti vlastiti primarni ključ koji je potreban za autorizaciju prema servisu, slanje svih 784 vrijednosti piksela (lista stringova). Zatim slijedi rukovanje odgovorom servisa pri čemu se prvo provjerava je li odgovor uspješan i bez ikakvih grešaka u komunikaciji između aplikacije i servisa. Ako je sve u redu onda se dohvaća rezultat klasifikacije i ispisuje unutar aplikacije u suprotnom ispisuje se greška koja je nastala.

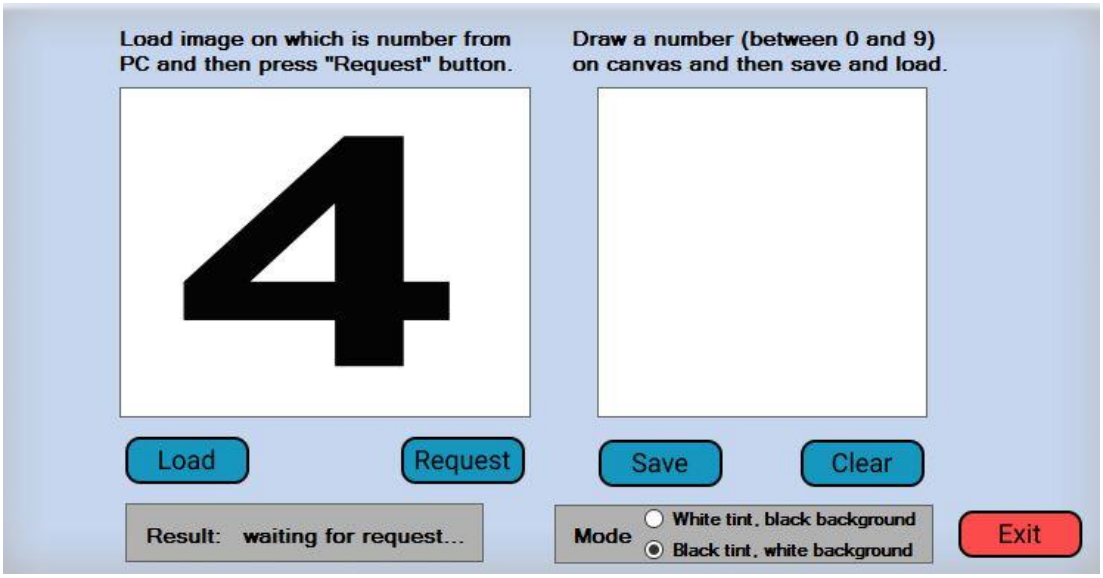
2.3. KORISNIČKA APLIKACIJA

Ova aplikacija je jednostavna za korištenje i napravljena je u C# Windows Forms. Kada se aplikacija pokrene otvara se glavni prozor aplikacije (Sl. 2.3.1.). Glavni prozor sadrži jedan PictureBox element i Panel iznad kojih se nalazi tekst koji služi kao kratka uputa što korisnik treba napraviti i kako se koristi aplikacija. PictureBox služi za prikaz učitane slike sa računala. Ispod se nalaze dva gumba: Load i Request. Load služi za učitavanje slike (podržava .jpg ili .png ekstenziju slike), a Request za slanje zahtjeva (vrijednosti piksela sa slike) prema servisu.



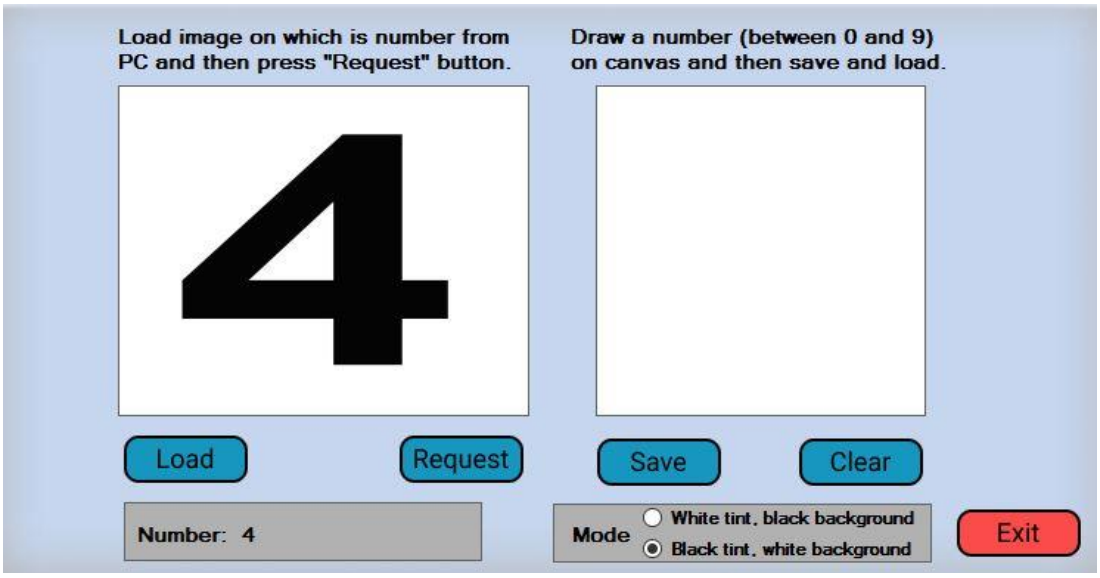
Sl. 2.3.1. Prikaz glavnog prozora aplikacije

Ukoliko učitana slika nije crno-bijela ispisuje se odgovarajuća poruka jer je model strojnog učenja namijenjen za prepoznavanje brojeva sa crno-bijelih slika, a ne u boji. Nakon pritisnutog gumba Request ubrzo se ispisuje rezultat u lijevom kutu sučelja aplikacije. S druge strane, Panel predstavlja platno na koje se može crtati pomoću miša. Svrha tog platna je da korisnik vlastoručno nacrta broj i zatim ga spremi. Ispod se također nalaze dva gumba: Save i Clear. Prvi služi za spremanje slike na računalu, a drugi za brisanje slike ukoliko korisnik nije zadovoljan kako izgleda broj ili želi nacrtati novi broj. Nakon što je spremio broj odmah može učitati taj isti broj i poslati zahtjev servisu pri čemu će se proći kroz model za klasificiranje te će biti vraćen rezultat predviđanja broja. Na slikama koje slijede može se vidjeti kako izgleda učitavanje slike, prikaz rezultata, grešaka i crtanje vlastitog broja.

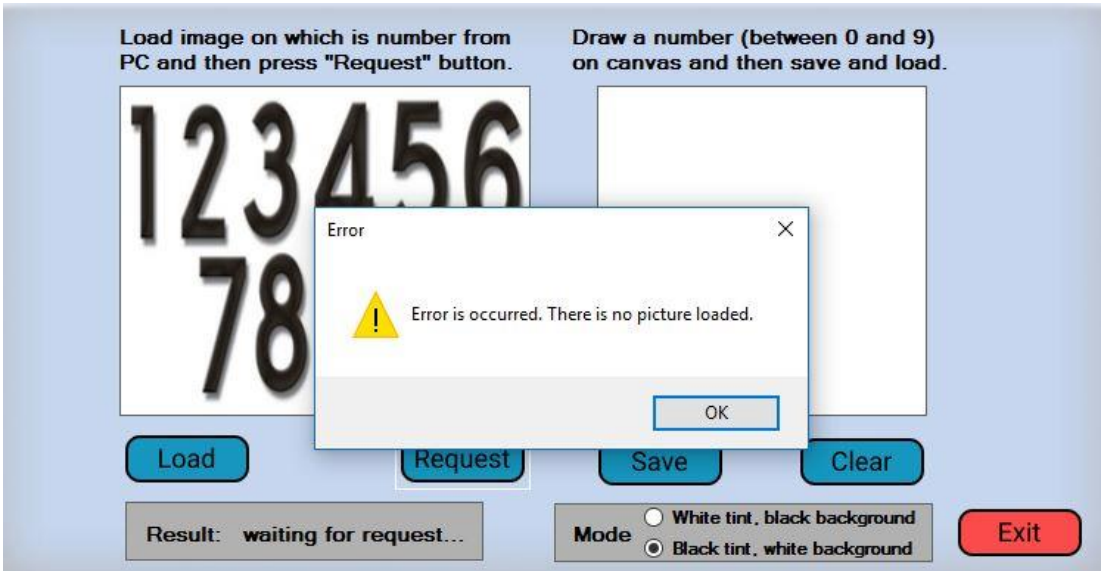


Sl. 2.3.2. Prikaz nakon učitanog broja

Nakon što je broj učitan idući korak je pritisnuti gumb Request kako bi vrijednosti piksela sa slike poslale servisu. Možemo vidjeti da se u donjem lijevom kutu nalazi tekst Result i da je ispisano da se čeka na slanje zahtjeva. Također je bitno uočiti da se mogu slati dva tipa slike: bijela pozadina s crnom tintom broja ili crna pozadina s bijelom tintom broja. Razlog takve podjele je zbog korištenog skupa podataka za treniranje koji imaju crnu pozadinu i bijelu boju broja. Ukoliko ne odaberemo biti će samo postavljeno na crnu tintu i bijelu pozadinu kao što je prikazano i na slici (Sl. 2.3.2.).

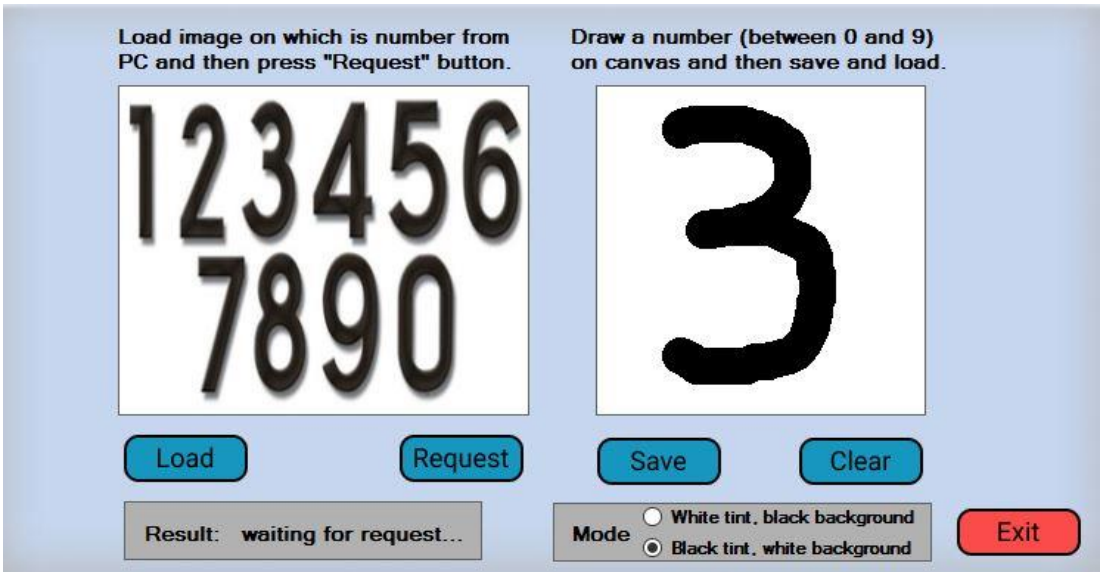


Sl. 2.3.3. Prikaz rezultata nakon pritisnutog gumba Request

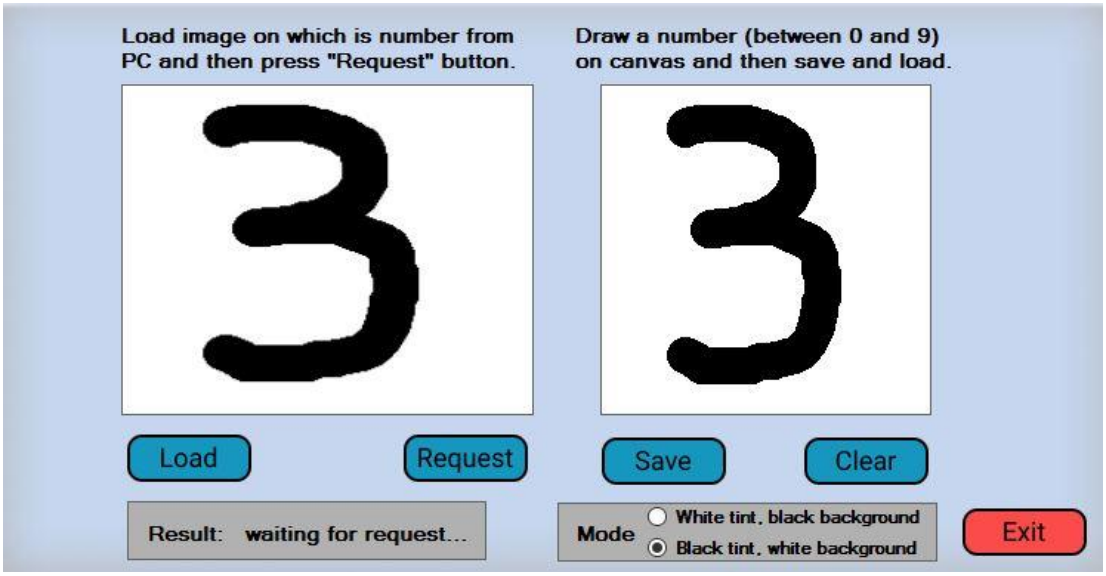


Sl. 2.3.4. Ispis greške prilikom pokušaja slanja zahtjeva bez prethodno učitane slike

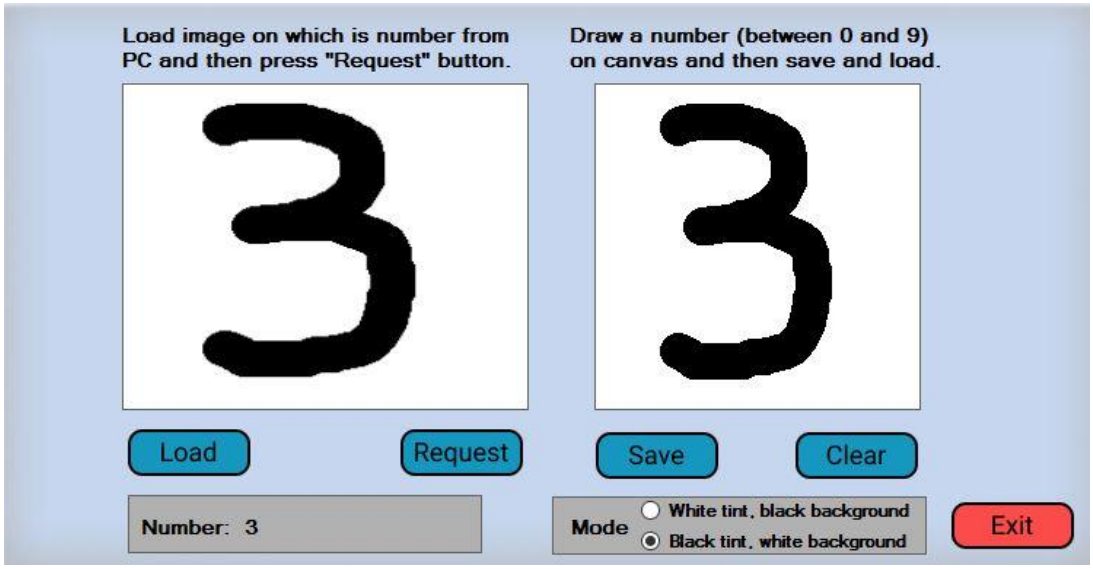
Slika iznad prikazuje pojavljivanje poruke s greškom ukoliko korisnik šalje zahtjev pri čemu nije prethodno učitana neka slika. Možda izgleda da postoji slika s brojevima od 0 do 9 unutar PictureBox-a, ali to je samo pozadinska slika (placeholder) koja nije aktivna i prikazana je čim se pokrene aplikacija. Razlog je tome kako bi korisnik bio upućen gdje će se prikazivati učitana slika. Sada slijede slike s prikazom broja koji je crtan unutar aplikacije, učitan i zatim poslan zahtjev servisu.



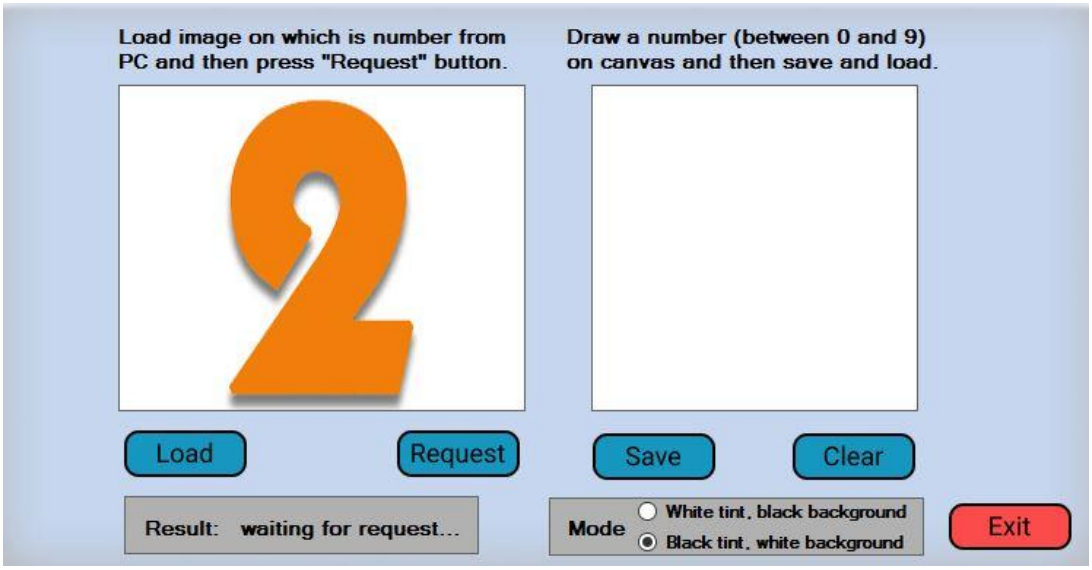
Sl. 2.3.5. Prikaz nacrtanog broja (desna strana) koji je zatim spremljen na računalo



Sl. 2.3.6. Prikaz nacrtanog broja koji je zatim i učitan (lijeva strana)

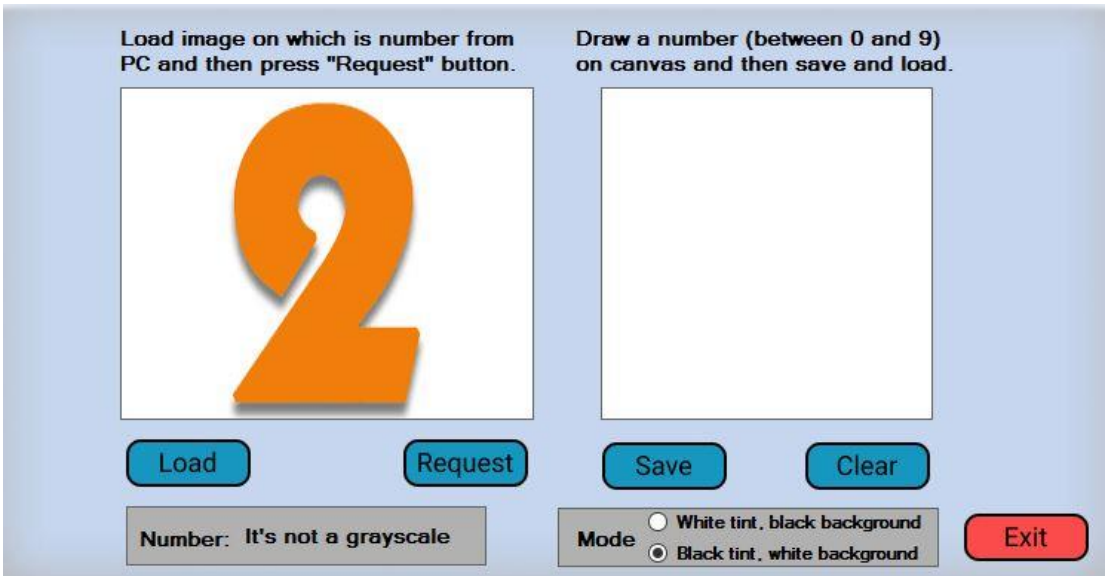


Sl. 2.3.7. Prikaz predviđenog broja, rezulata nakon pritisnutog gumba Request

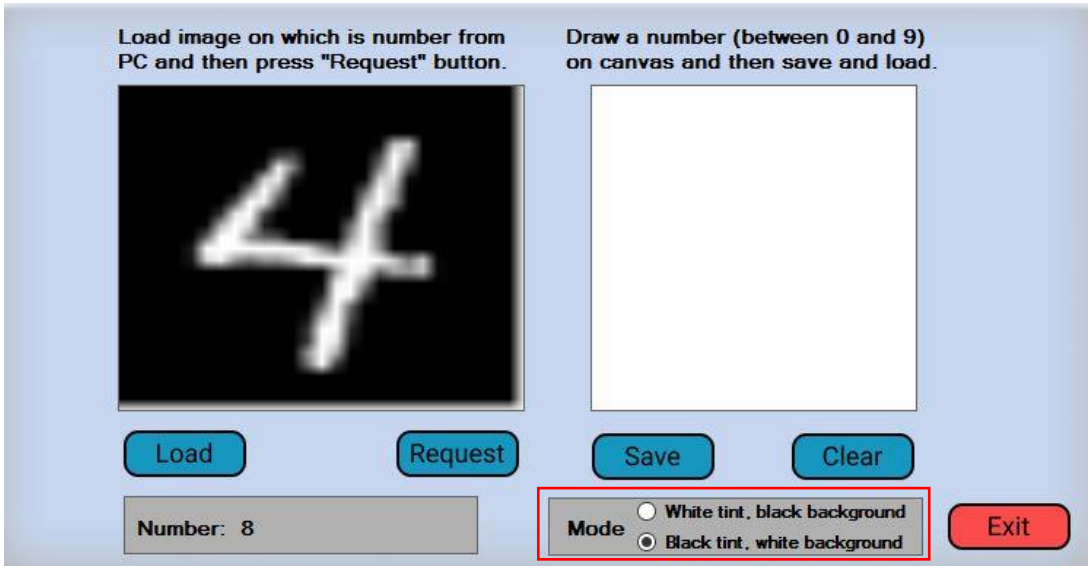


Sl. 2.3.8. Učitana slika koja je u boji, a predstavlja broj

Slika iznad predstavlja broj 2, ona je u boji te njezini pikseli stoga neće biti prosljeđeni na servis. Razlog je taj što su podaci na kojima je treniran model slike crno-bijele boje i ne bi mogao dobro klasificirati broj. Za dani primjer sa slike model bi prepoznao kao broj 8 i to s vjerojatnošću preko 60% što nikako nije točno. Ispod se nalazi rezultat dobiven nakon što je poslan zahtjev.

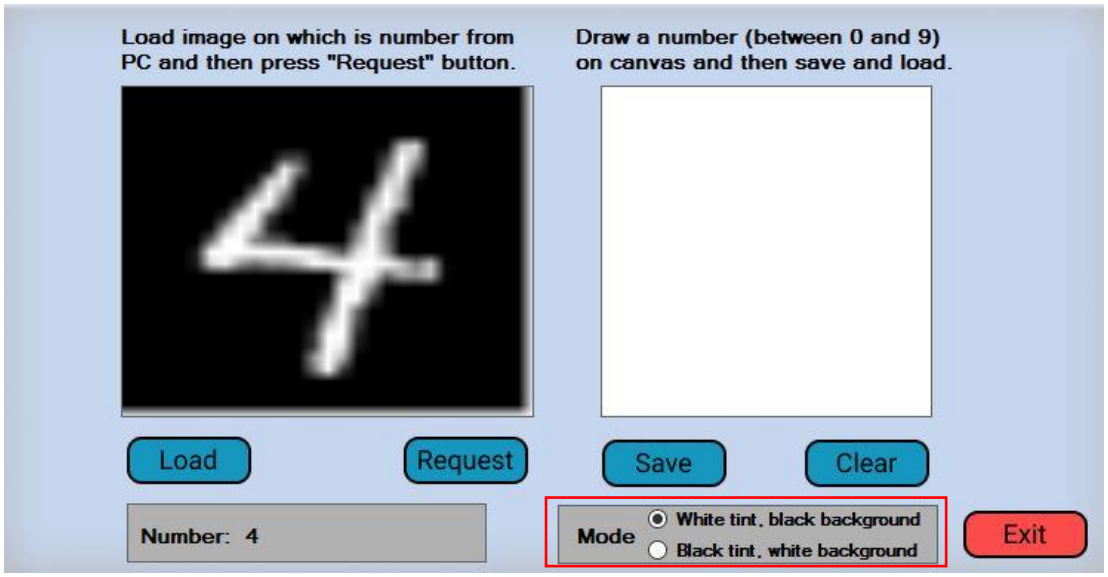


Sl. 2.3.9. Rezultat slike javlja porukom da nije crno-bijela



Sl. 2.3.10. Rezultat slike koja je crno-bijela, ali je krivi tip slike označen

Slika iznad predstavlja primjer kada je slika uistinu crno-bijela, ali je postavljen krivi tip slike. Zbog toga neće doći do nikakve poruke o grešci, ali će broj sa slike biti krivo klasificiran. Korisnik, da bi izbjegao takav ishod može odabrati drugi tip slike gdje se gleda da je crna pozadina i bijela tinta.



Sl. 2.3.10. Rezultat slike kada je označen ispravan tip slike

Možemo vidjeti da je rezultat ispravno klasificiran broj, pri čemu je za broj sa slike predviđeno da se nalazi četvorka s vjerojatnošću od 81%, što nije nimalo loš rezultat.

3. ZAKLJUČAK

Na temelju modela i rezultata koji su dobiveni i ostvareni u ovom projektu za klasificiranje brojeva sa slike možemo reći da to nije jednostavan zadatak i da je gotovo nemoguće postići da je rezultat uvijek točan. Kao gotovi model s algoritmom klasificiranja, pri čemu je korištena neuronska mreža pokazalo se da s vrlo velikom točnošću (od 97.82%) se može predvidjeti koji je broj na slici iz korištenog skupa podataka za treniranje i testiranje. Bitno je napomenuti da na točnost i kvalitetu modela jako utječe skup podataka koji se koristi za treniranje. Ukoliko je skup podataka nedovoljno generaliziran, prilikom primjenjivanja takvog istreniranog modela neće se postići zadovoljavajući rezultati na stvarnim podacima. Pošto se za ovaj model klasificiranja oslanja na vrijednost piksela sa slike jako je teško postići takvu generalizaciju skupa podataka koja bi omogućila da istreniran model prepozna točno svaki broj sa slike. Koristeći Microsoft Azure Machine Learning Studio vidimo da poprilično brzo i jednostavno možemo napraviti model koji ćemo moći istrenirati i testirati bez da moramo praviti vlastiti servis. Sve to uvelike olakšava posao jer se sve izvršava na cloud-u te nije potrebno instalirati nikakav dodatan program za rad s ovakvim alatom. Tako je uz gotovi web servis jedino potrebno napraviti vlastitu aplikaciju kao sučelje prema korisniku kako bi lako i jednostavno koristio web servis. Ovakav model mogao bi biti koristan i proširiv za razne primjene poput snimljenih prometnih znakova putem kamere pri čemu bi se izdvajale slike sa znakovima koji predstavljaju ograničenje brzine. Zatim bi se takve slike dalje obrađivale i izdvajali brojevi te slali računalu na obradu. Na temelju prikupljenih informacija donosila bi se odluka za daljnje upravljanje vozilom na temelju ograničenja koji je definiran znakom i trenutne brzine vozila.

4. LITERATURA

[1] Handwritten Numbers, data set

https://www.kaggle.com/sarahstory/predicting-handwritten-numbers/data

[2] Multiclass Decision Forest, machine learning

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-decision-forest

[3] Support Vector Machine (SVM), machine learning

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-support-vector-machine

[4] Multiclass Decision Jungle, machine learning

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-decision-jungle

[5] Multiclass Neural Network, machine learning

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-neural-network

[6] EmguCV, image processing library

http://www.emgu.com/wiki/index.php/Main_Page

5. POVEZNICE I PRILOG

Programsko rješenje nalazi se na github repozitoriju koji je dostupan na linku ispod.

https://github.com/Ruap/Ruap-project---ML.git

U nastavku slijedi Python kod koji je korišten za rekonstrukciju slika na temelju test.csv dokumenta unutar kojeg se nalaze vrijednosti piksela za svaku sliku (Sl 5.1.).

```
import csv
import numpy as np
from PIL import Image

def ImageReconstruction(dataSet, savePath, header):

    data = list(csv.reader(open(dataSet)))
    if(header):
        data = data[1:] #ukoliko ima hedera da se izbaci za konstrukciju
slike/slika
    try:
        for i in range(len(data)):
            rez = np.array(data[i]).reshape((28,28)) #spremanje podataka iz
cijelog retka u polje/matricu 28x28
            array = np.array(rez, dtype=np.uint8)
            new_image = Image.fromarray(array)
            new_image.save(savePath + str(i) + '.png')
    except:
        print("Check function arguments, especially header value (True/False)?")

ImageReconstruction('test.csv', 'C:/Users/Ivan/Desktop/Testne slike/',
header=True)
```

Sl. 5.1. Python kod za rekonstrukciju slika