**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEK**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I**

**INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni studij**

**RASPOZNAVANJE RUKOM PISANIH ZNAKOVA**

**Seminarski rad**

**David Turkalj, David Hodak**

**Osijek, 2020.**

**Sadržaj**

[**1. UVOD 1**](#_Toc64489143)

[**2. TIJEK IMPLEMENTACIJE PROJEKTA 2**](#_Toc64489144)

[**2.1. Korišteni podaci 2**](#_Toc64489145)

[**2.2. Azure machine learning 2**](#_Toc64489146)

[**2.3. Treniranje modela 3**](#_Toc64489147)

[**2.3.1. Multiclass Neural Network 3**](#_Toc64489148)

[**2.3.2. Multiclass Decision Forest 6**](#_Toc64489149)

[**2.3.3. Multiclass Logistic Regression 8**](#_Toc64489150)

[**2.3.4. Osvrt na treniranje 9**](#_Toc64489151)

[**3. KLIJENTSKA APLIKACIJA 10**](#_Toc64489152)

[**3.1. Problemi 13**](#_Toc64489153)

[**4. ZAKLJUČAK 14**](#_Toc64489154)

[**5. POVEZNICE I LITERATURA 15**](#_Toc64489155)

# UVOD

U ovom projektu razrađuje se popularni problem raspoznavanja rukom pisanih znakova. S obzirom da svaki čovjek ima sebi svojstven rukopis, pokušava se naučiti računalo da prepozna znakove različitih veličina i oblika.

Od znakova koriste se samo velika slova engleske abecede. Cijeli sustav obrađuje se na Azure Machine Learning platformi koja se nalazi u oblaku. Podaci se razmjenjuju preko web servisa.  
Za klijentsku aplikaciju i testiranje treniranog modela implementirana je Windows desktop aplikacija koristeći C# i Windows Forms framework.

Kako je ovo problem klasifikacije korišteno je nekoliko metoda strojnog učenja te je odabrana ona s najboljim rezultatima za klijentsku aplikaciju u kojoj će se testirati na stvarnim podacima.

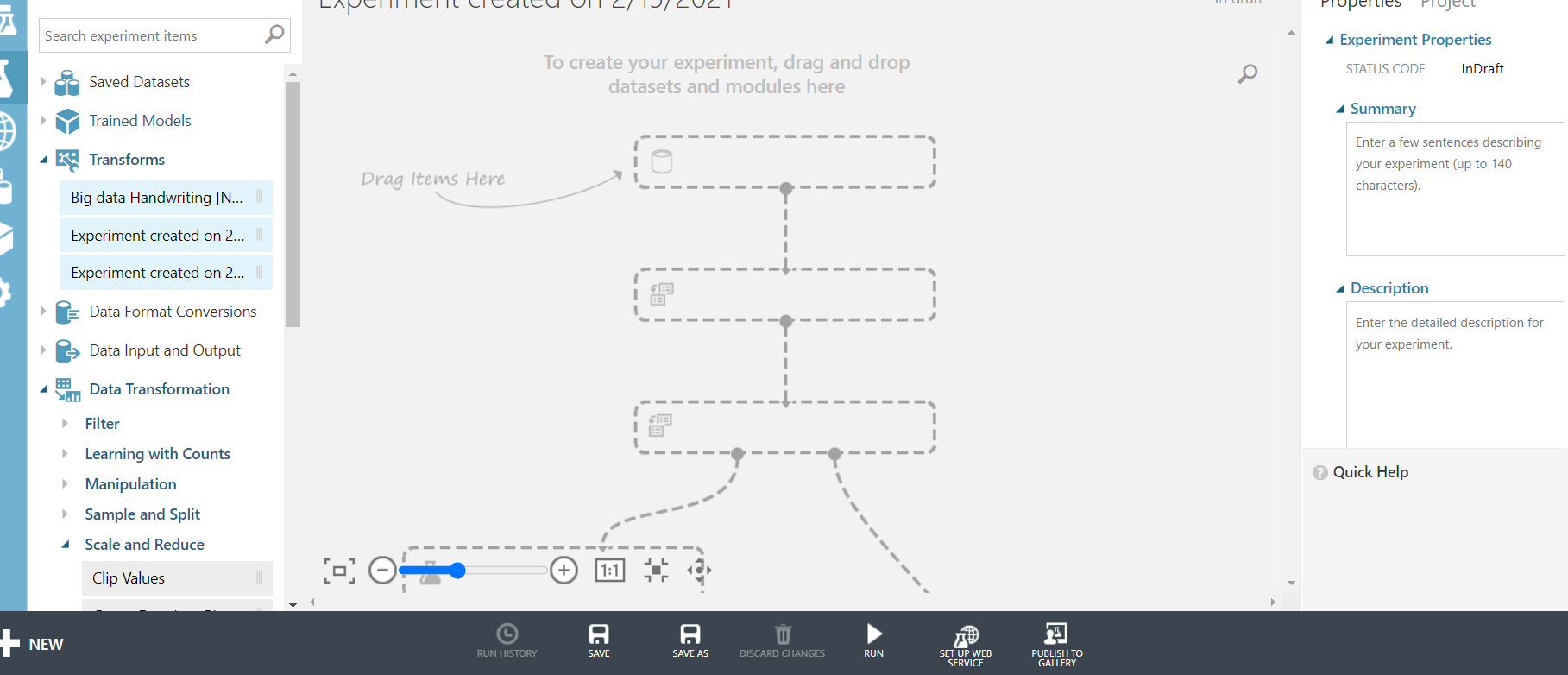
# TIJEK IMPLEMENTACIJE PROJEKTA

## ****Korišteni podaci****

Podaci su preuzeti s Kaggle web stranice u obliku csv datoteke. Slike slova prebačene su u csv tip datoteke tako da svaki redak predstavlja jednu sliku s 785 stupaca gdje je prvi stupac klasifikacijska vrijednost od 0 do 25(26 slova engleske abecede), a ostalih 784 sadrži vrijednost piksela od 0(crna) do 255(bijela). Veličina slike je 28x28 piksela. Ova baza podataka sastoji se od 370 tisuća slika. Ovakav izbor značajki može izgledati kao „brute force“ način prikupljanja podataka, ali je jednostavan i brz način prikupljanja uz pomoć alata za obradu slika te ne zahtjeva komplicirane matematičke analize svake od slika. Ono što se može odmah zaključiti je da će proces predobrade i treniranja modela biti vremenski intenzivan s obzirom na količinu podataka.

## ****Azure machine learning****

**Azure machine learning je okolina u oblaku koja se koristi za treniranje, razvijanje, automatiziranje, upravljanje i praćenje modela strojnog učenja. Može se koristiti za bilo koji oblik strojnog učenja, od klasično do dubokog, nadziranog i nenadziranog učenja. Mogućnost je rada s programskim jezicima Python i R ili pak za one koji zaziru od pisanja koda tu je veoma intuitivan sustav povlačenja i ispuštanja (eng. *drag and drop*) za razvijanje cijelog procesa strojnog učenja od dohvaćanja i obrade podataka, treniranja modela te evaluacije. Microsoft nudi i kreiranje besplatnog računa s kojim je moguće koristiti široki spektar mogućnosti Azure ML-a.**

****

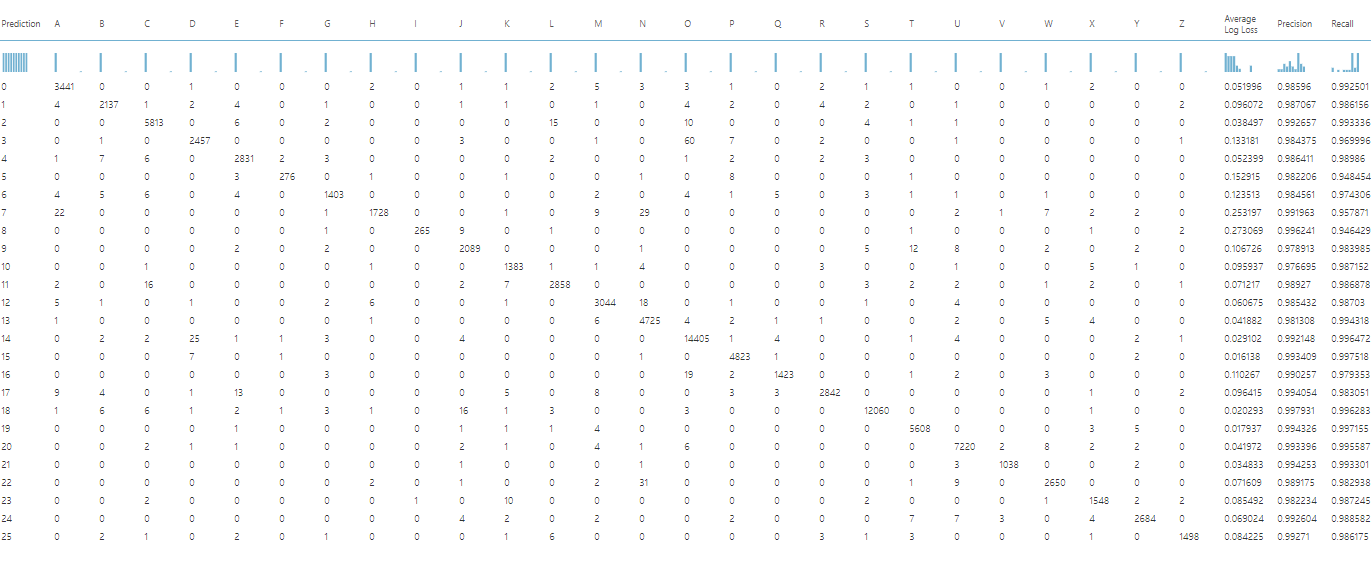
**Slika 2.1. *Izgled okoline Azure Machine Learninga***

## ****Treniranje modela****

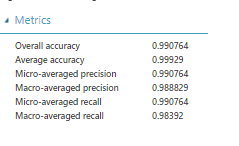
U ovom projektu koristiti se više metoda treniranja. Prije nego što ih uključimo u eksperiment potrebno je ubaciti skup podataka te ga razdijeliti u podskupove za trening i testiranje. Razlog tomu je što želimo da se trenirani model testira na podacima koje nije vidio, odnosno koji nisu korišteni za treniranje. U suprotnom bi moglo doći do prekomjernog prilagođavanja podataka.   
Metode koje se koriste su Multiclass Neural Network, Multiclass Decision Forest i Multiclass Logistic Regression. Sve tri metode su zapravo namijenjene za ovaj problem te se konačni rezultat razlikuje za male margine.

* + 1. **Multiclass Neural Network**

Model neuronske mreže koji se koristi kod predviđanja podatka s više vrijednosti pa je prigodno odabrati ovakav model za prepoznavanje slova. Klasifikacija korištenjem neuronskih mreža je nadzirano učenje pa se očekuje da baza podataka ima stupac s oznakom klase. Neuronska mreža je skup međusobno povezanih slojeva koji se sastoji se od neurona. Oni primaju određeni broj podataka (x1, x2,…) koji se množe sa specifičnom težinom(w1,w2…). Zatim se takvi podaci zbrajaju ( z = ) i predaju aktivacijskoj funkciji koja kao produkt ima izlaz y = f(z).  
Mreža se sastoji od jednog ulaznog sloja, koji prima ulazne podatke, a zatim se izvršavaju proračuni na svakom neuronu te se izlazni podaci prosljeđuju na iduće slojeve, skrivene slojeve i na kraju na završni sloj koji daje gotovi rezultat. Moguće je imati nijedan skriveni sloj, ali i mnogo njih. Više skrivenih slojeva omogućava rješavanje kompleksnijih problema kao što je i prepoznavanje slova ili brojeva sa slika. Za problem prepoznavanja slova izabran je jedan skriveni sloj i 1000 neurona u njemu. Svi ostali parametri su ostali predefinirani.



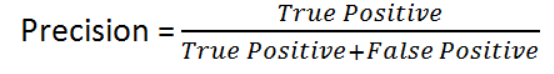
**Slika 2.2.** *Matrica konfuzije*



**Slika 2.3.** *Statistika modela nakon treniranja*

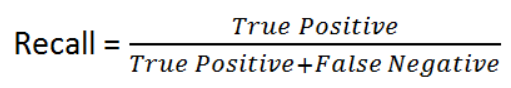
Prema slici 2.2. možemo reći kako je ovo dobar model i to ne samo prema točno predviđenim podacima već i prema recall i precision vrijednostima.

Precision rate prikazuje koliko je model točan ako uzmemo u obzir koliko od onih predviđenih pozitivno je zapravo pozitivno.



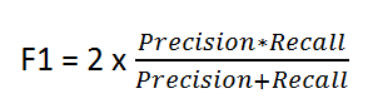
**Slika 2.4.** *Formula precision vrijednosti*

Recall rate prikazuje koliko je naš model od onih stvarno pozitivnih predvidio kao pozitivne.



**Slika 2.5.** *Formula recall vrijednosti*

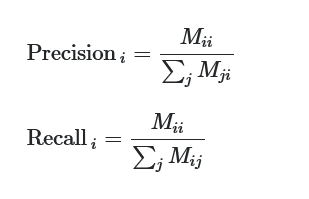
F1 Score je pokazatelj ravnoteže između precision rate i recall rate te je vjerniji pokazatelj točnosti modela nego accuracy rate, posebice ako su podaci nejednako distribuirani(veliki broj stvarno negativnih - False Positive + True Negative).

****

**Slika 2.6.** *Formula F1 vrijednosti*

Kako bi generalizirali ove formule za više klasne probleme prema slici 2.7. možemo reći da je precision vrijednost gdje smo točno predvidjeli (i) podijeljeno s zbrojem svih onih vrijednost gdje je algoritam predvidio (i).

Recall s druge strane je vrijednost gdje smo točno predvidjeli (i) podijeljeno s zbrojem svih vrijednosti gdje (i) zapravo ima pravo stanje



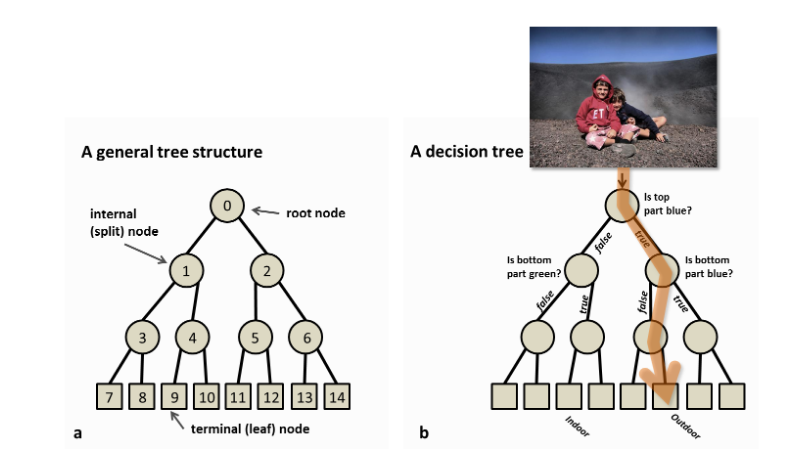
**Slika 2.7.** *Formula za više klasa.*

Accuracy rate nije najbolji pokazatelj točnosti jer postoje situacije u kojima pozitivno predviđeni podaci spadaju u kategoriju koja u realnosti znači nešto negativno(krađa, bolest itd) pa tako model koji ima točnost 98%, može svrstati par podataka koji su pozitivni u negativne, a radi se o bolesnicima s visoko zaraznom bolešću.   
 Još jedna bitna statistika je važna za probleme klasifikacije, prikazana je na slici 2.3, a to su makro i mikro prosjeci. Makro prosjek će izračunati metriku za svaku klasu, a zatim uzeti prosjek što nam govori da tretira sve klase jednako. Mikro prosjek će prikupiti doprinose svih klasa(podatke točno previđenih i netočno predviđenih) za izračunavanje prosječne metrike. Ukoliko se sumnja na neravnotežu u broju podataka po klasama mikro prosjeci su važniji.   
U ovom projektu postoji slučaj ne balansiranih klasa što ćemo vidjeti u kasnijem dijelu kod testiranja klijentske aplikacije i zašto se model muči prepoznati primjerice, slovo I. Dok za recimo slovo T postoji preko 10000 slika, za slovo I postoji samo 1100. Ne balans se može vidjeti i u tablici konfuzije. To može utjecati na prepoznavanje modela slova koja imaju sličnu strukturu.

Ipak, sve statistike pokazuju da je model neuronske mreže i više nego zadovoljio s preciznošću i kao takav je odabran za testiranje preko klijentske aplikacije.

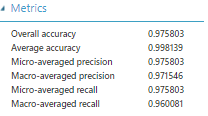
* + 1. **Multiclass Decision Forest**

Stablo odluke je skup neurona i bridova organiziranih u hijerarhijsku strukturu. Čvorovi(neuroni) su podijeljeni u unutarnje i vanjske čvorove. Unutarnji se označavaju s krugovima, a vanjski s kvadratima što se može vidjeti na slici 2.8. Stablo ne sadrži petlje, svi čvorovi imaju jedan ulaz. Ovakva stabla se koriste za donošenje odluka. Tijekom donošenja odluka unutarnji čvorovi primjenjuju funkcije na podatke te ih šalju na odgovarajući pod čvor(eng. *child node*). Zadatak u kojem se ova metoda koristi je klasifikacija. Razlog tome je što se dobro nosi s problemima od više klasa. Pruža podatke o vjerojatnosti. Dobro generalizira prethodno ne viđene podatke te je iznimno učinkovita zahvaljujući paralelizmu i smanjenom broju testova po pojedinačnom podatku.



**Slika 2.8.** *Struktura stabla odluke*

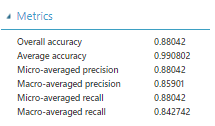
Prikazom statistike na slici 2.9. za decision forest metodu zaključujemo da je ona vrlo moćan alat za probleme klasifikacije. Za nijansu je lošija od neuronske mreže iako s povećanjem grana i dubinom, rezultat bi zasigurno bio bolji.



**Slika 2.9.** *Statistika modela nakon treniranja*

* + 1. **Multiclass Logistic Regression**

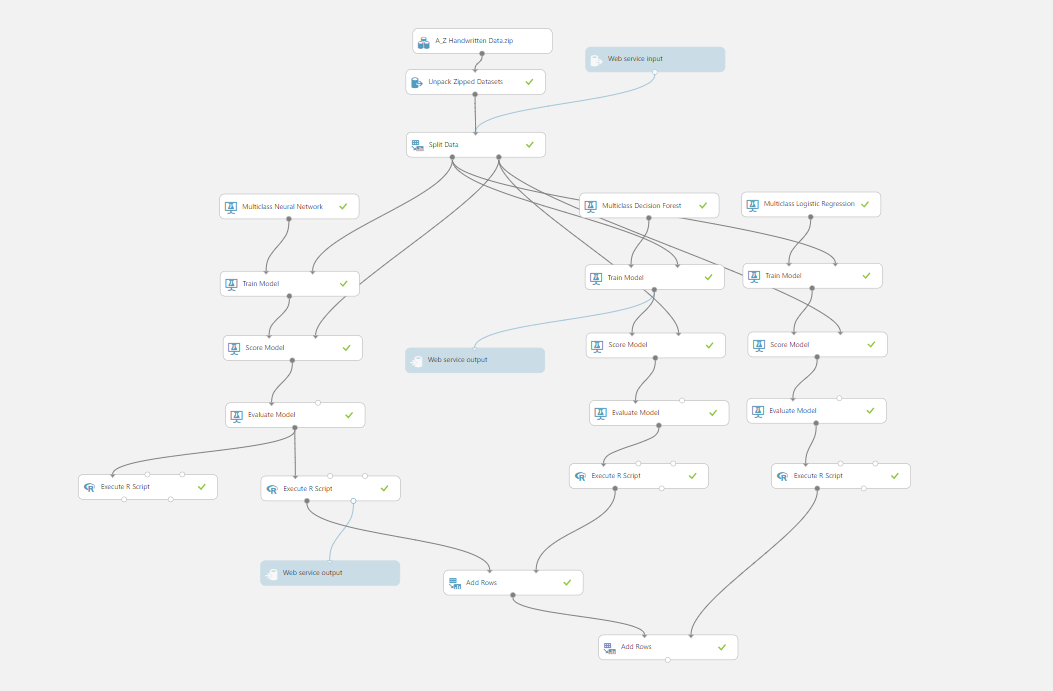
Dodatak na klasičnu logističku regresiju koja se koristi za klasifikaciju problema s dvije klase, ova metoda može klasificirati mnogo više. Logistička regresija koristi binomnu funkciju raspodjele vjerojatnosti te mapira oznake s 1 za pozitivnu klasu, a 0 za negativnu. Više klasna logistička regresija se koristi kada je ovisna varijabla nominalna(ekvivalentno kategorična, odnosno spada u bilo koju kategoriju koja se ne može na smisleni način kategorizirati) i kada postoji više od dvije kategorije. Primjerice, kojoj krvnoj grupi pripada čovjek. Jedan od principa prilagođavanja logističke regresije problemima klasifikacije s više klasa jest razdvajanje problema na više problema s binarnom klasifikacijom i uklapanje standardnog modela logističke regresije. Alternativni pristup mijenja model logističke regresije tako da izravno podržava predviđanje više klasa, predviđajući vjerojatnosti da ulazni podatak pripada svakoj od poznatih klasa.



**Slika 2.10.** *Statistika modela nakon treniranja*

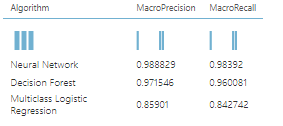
Iako su stabla odluke i neuronske mreže bolji i efektivniji alati, više klasna logistička regresija ne pokazuje toliko lošije rezultate.

* + 1. **Osvrt na treniranje**



**Slika 2.11.** *Izgled eksperimenta*

Azure Machine Learning je uveliko ubrzao cijeli proces te posebice dio s računalnim performansama jer se sve radi u oblaku, dok korisniku omoguće da nesmetano dalje istražuje i koristi računalo. Korištena baza podataka slova je velika te je cijeli proces znao potrajati i do 20 minuta.



**Slika 2.12.** *Konačna statistika*

S obzirom na rezultate neuronska mreža je odabrana za postavljanje na web servis, odnosno koristi se pri testiranju slika koje se generiraju iz klijentske aplikacije.

# KLIJENTSKA APLIKACIJA

Aplikacija je izrađena uz pomoć Windows Forms biblioteke koja sadrži grafičke klase otvorenog koda kao dio Microsoft. NET Frameworka. Prvotno je kreiran jedan PictureBox element koji će poslužiti kao platno po kojem će se pisati mišem. Kod koji opisuje funkcionalnost pisanja mišem po PictureBox elementu prikazan je na slici 3.1. Ono što je bitno naglasiti je da se zapravo

private void canvas\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = e.Location;

isMouseDown = true;

}

private void canvas\_MouseMove(object sender, MouseEventArgs e)

{

if (isMouseDown == true)

{

if (lastPoint != null)

{

submitBtn.Enabled = true;

if (canvas.Image == null)

{

Bitmap bmp = new Bitmap(canvas.Width, canvas.Height);

canvas.Image = bmp;

}

using (Graphics g = Graphics.FromImage(canvas.Image))

{

pen.StartCap = pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

g.DrawLine(pen, lastPoint, e.Location);

}

canvas.Invalidate();

lastPoint = e.Location;

}

}

}

private void canvas\_MouseUp(object sender, MouseEventArgs e)

{

isMouseDown = false;

lastPoint = Point.Empty;

}

**Slika 3.1*.*** *Prikaz koda konfiguracije platna za pisanje*

piše po slici koja je kreirana kao bitmap tip podatka po veličini PictureBox elementa. Na pritisak gumba Submit(slika 3.2.) ta se slika sprema te ponovno poziva i smanjuje na onu veličinu koju imaju sve ostale slike iz baze podataka korištene za treniranje modela. To nam omogućava da očitamo sve dostupne piksele i njihove vrijednosti te ih zapišemo u listu koja će se kasnije predati kao json input mreži treniranoj na Azure ML platformi.

private void submitBtn\_Click(object sender, EventArgs e)

{

submitBtn.Enabled = false;

probabilities.Clear();

ParseDataToWebService();

resultBtn.Enabled = true;

}

void ParseDataToWebService()

{

int counter = 1;

canvas.Image.Save("canvas.bmp");

Bitmap image = new Bitmap("canvas.bmp");

Bitmap resized = new Bitmap(image, new Size(28, 28));

resized.Save("resized.bmp");

List<Dictionary<string, string>> jsonList = new List<Dictionary<string,string>>();

Dictionary<string, string> word = new Dictionary<string, string>();

word.Add("Col1", "0");

for (int x = 0; x < 28; x++)

{

for (int y = 0; y < 28; y++)

{

counter++;

word.Add(("Col" + counter), resized.GetPixel(y, x).A.ToString());

}

}

jsonList.Add(word);

image.Dispose();

InvokeRequestResponseService(jsonList).Wait();

}

**Slika 3.2**. *Prikaz koda za obradu podataka slike*

Azure nam srećom pomaže pri povezivanju web servisa s aplikacijskim dijelom. Nudi generirani kod u C#, python i R programskim jezicima. S obzirom da korištena baza podataka ima 785 stupaca toliko će biti potrebno i inputa poslati u json formi preko aplikacije. Odmah je jasno da će takav format izgledati nepregledno, no uz malo programerske magije i taj je problem riješen. Kreirana je lista rječnika kojoj se predaje poseban rječnik stringova u koji se po izgledu konstrukcije stupaca baze podataka popunjavaju imena i vrijednosti stupaca. Zatim se ta lista predaje funkciji InvokeRequestResponseService. Ona je zadužena za komunikaciju s Azure ML preko posebnog api ključa i uri adrese. Važno je naglasiti da se podaci moraju predati u

static async Task InvokeRequestResponseService(List<Dictionary<string, string>> jsonList)

{

using (var client = new HttpClient())

{

var scoreRequest = new

{

Inputs = new Dictionary<string, List<Dictionary<string, string>>>() {

{

"input1",

jsonList

},

},

GlobalParameters = new Dictionary<string, string>()

{

}

};

const string apiKey = "U31tpjuwO73Et6728s4lWdBTE66NBPnDBIVJUy8ZKpcGY0IRMU+6+q0CKMAq3ZDw4yTpDR0Dk8dJ2wvceYczDg==";

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);

client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/5dca254c248b478aa3953e9b7b25ba3f/services/491546f09a004c2784d9d8eb7a4201cf/execute?api-version=2.0&format=swagger");

HttpResponseMessage response = await client.PostAsJsonAsync("", scoreRequest).ConfigureAwait(false);

if (response.IsSuccessStatusCode)

{

string result = await response.Content.ReadAsStringAsync();

GetPredictionValues(result);

}

else

{

Debug.WriteLine(string.Format("The request failed with status code: {0}", response.StatusCode));

Debug.WriteLine(response.Headers.ToString());

string responseContent = await response.Content.ReadAsStringAsync();

Debug.WriteLine(responseContent);

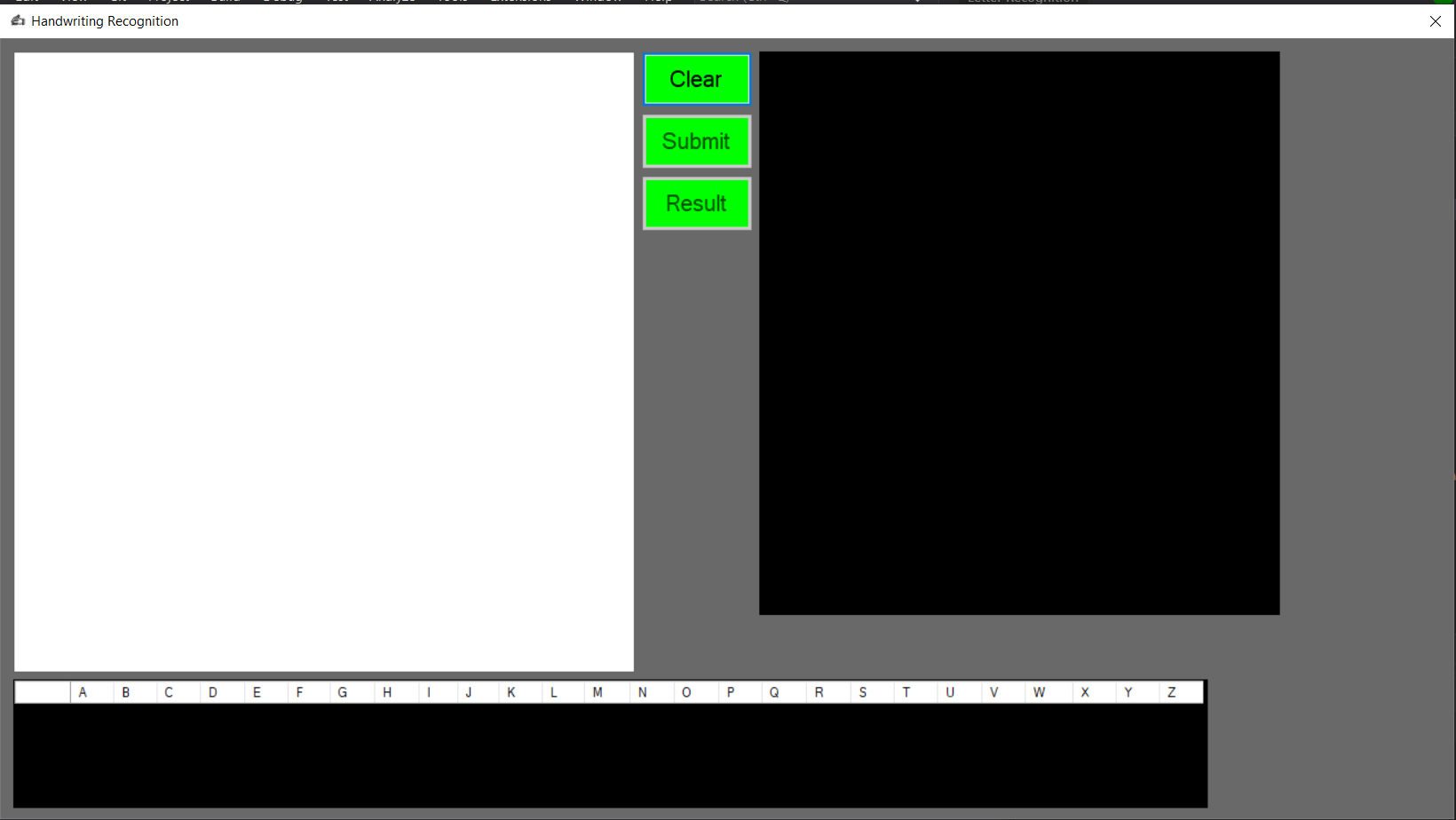
}

}

}

**Slika 3.3.** *Pristupanje Azure web servisu*

ispravnoj json formi, da su api ključ i adresa ispravni. Ako je sve u redu povratno ćemo dobiti podatke o predviđanju za svaku klasu i onu oznaku klase za koju se smatra da joj pripadaju podaci. GetPredicitonValues je dodatno napisana funkcija kojom se ti podaci preuređuju i “izvlače” jer dolaze u jednom stringu.



**Slika 3.4.** *Izgled aplikacije*

Izgled aplikacije je jednostavan i minimalistički. Lijevo platno je zaduženo za pisanje, desno će se prikazati koje slovo je model predvidio ako pritisnemo Submit pa Result gumb. U donjoj tablici će pisati vjerojatnosti po klasama, odnosno slovima.

* 1. Problemi

S obzirom da je model neuronske mreže dao skoro savršene rezultate pretpostavilo bi se da će i za bilo koju drugu sliku slova uvijek dati točan rezultat. No to baš i nije tako. Model je dao izvrsne rezultate na slikama koje su specifično kreirane za tu bazu podataka. Slova su približno iste debljine i veličine te nalaze se uvijek oko centra. Istu stvar moramo primijeniti i u aplikaciji. Ako napišemo slovo u gornjem desnom kutu naravno da nećemo dobiti dobar rezultat. Debljina linije kojom pišemo također ima velik utjecaj. Odabrana je neka optimalna vrijednost za debljinu koja je dala najbolje rezultate i koja izgleda približno onoj debljini slova iz slika u bazi podataka, kada se smanji na veličinu 28x28 piksela. Mora se naglasiti da na slici s aplikacije nije odrađena nikakva obrada (Gaussovo zamućivanje, detekcija rubova itd.) već je smanjena na traženu veličinu i očitane su vrijednosti piksela koje su doduše u rasponu od 0 do 255 jer se koristi crna boja linije.

# ZAKLJUČAK

Prepoznavanje rukom napisanih znakova je opće poznati problem i služi za uvod u svijet strojnog učenja. Postoji mnogo baza podataka i metoda za klasifikaciju ovog problema te su neke od njih korištene i u ovom projektu. Azure Machine Learning platforma pokazala se kao veoma koristan alat u obradi podataka i treniranju modela te je uveliko olakšala cijeli proces projekta. Sve tri metode su pokazale zadovoljavajuće rezultate s naglaskom na neuronsku mrežu. Kako se ne bi zadržali samo na statistikama kreirana je i aplikacija koja povezuje klijenta s pozadinskim modelom koji se na taj način testira s podacima iz stvarnog svijeta.

# POVEZNICE I LITERATURA

[1] N. Buduma, Fundamentals of Deep Learning, str.7 i str.245-248, O'Reilly Media Inc., Sjedinjene Američke Države, 2017.

[2] How deep learning works, https://www.mathworks.com/discovery/deep-learning.html,

[3] <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-decision-forest>

[4]<https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2>

[5]<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/decisionForests_MSR_TR_2011_114.pdf>

[6] <https://machinelearningmastery.com/multinomial-logistic-regression-with-python/>

[7] <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-5.0>

[8] <https://en.wikipedia.org/wiki/Multinomial_logistic_regression>