Machine Learning-Based Handwriting Recognition

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Reema Domadia
Dinesh Padmanabhan
Utkarsh Nigam
DATS6202
Professor Amir Jafari
June 30, 2020

Table of Contents

Problem Statement:

The purpose of this project is to create a machine learning-driven handwriting recognition application that can assist doctors by continuously working to convert the handwritten script into a computerized legible script.

Introduction

Medication errors in hospitals are common, expensive, and sometimes fatal to patients. Doctors are infamously known for their sloppy scribbling in their handwritten prescriptions that lead to thousands of medication errors each year. A simple mistake can have serious consequences. Drugs with similar names such as the pain medication, Celebrex, and the antidepressant, Celexa, or the tranquilizer, Zyprexa, and the antihistamine, Zyrtec, amplify the importance of correctly decrypting handwriting. (Center for the Advancement of Health, 2007) Historically, the notes that doctors scribbled into a patient's medical history record file were generally only seen by doctors. However now, doctors are just one component of a multidisciplinary healthcare network. Illegible records composed by rushed doctors are now presented to colleagues with no qualifications in cryptology. Sixty-one percent of medication errors sustain injuries or die in hospitals as a result of illegible handwriting, according to a sciencedaily.com article titled “Computerized Doctors' Orders Reduce Medication Errors”. (Center for the Advancement of Health, 2007) The advantage of using an ML-driven handwriting recognition software is that the script can change multiple hands, and everyone will receive the same message, leaving no room for misinterpretation.

In addition to improving patient safety, computerized systems can also enhance the daily duties for professionals in the healthcare network as they would not have to spend time deciphering illegible handwriting. By creating less work for the health professional, not only are we decreasing stress on the individual, but we are also enabling them to be able to be more productive on their jobs by attending more patients or requests. This capability is crucial for industries like healthcare as medical records are often passed onto a web of professionals/ specialists, ranging from doctors to pharmacists, to complete the care of a patient. A minor miscommunication due to illegible handwriting care causes the patient their life.

The following sections of this report will illustrate our dataset, methodology, analysis, and the results. The dataset segment will explain the source of our dataset and explain exploratory data analysis finding that shaped the course of our project. The methodology highlights the model’s we’ve created and software’s we used to create the platform for our analysis. Finally, the analysis and results will showcase our findings and concluding remarks.

Dataset

The purpose of this research project is to bridge the capability gap in communication by training a model that will identify and decrypt handwriting to convert into a computer-typed text. While complete accuracy will be achieved over-time through continuously training the model with data samples, to address the scope of this prototype we will be using a dataset created by the National Institute of Standards and Technology (NIST). The NIST Special Database 19 consists of roughly 0.7 million sample png images serving as observations in our dataset. The abundance of data contained in this database will suffice to train our model. The following table will highlight the number of observations per character:

Table 1: *Number of Observations Per Characters*

A: 7,010	Q: 2,566	g: 3,839	w: 2,699
B: 4,091	R: 4,536	h: 9,713	x: 2,820
C: 2,792	S: 23,827	i: 2,788	y: 5,088
D: 4,945	T: 10,927	j: 1,920	z: 2,726
E: 5,420	U: 14,146	k: 2,562	0: 34,803
F: 10,203	V: 4,951	l: 16,937	1: 38,049
G: 2,575	W: 5,026	m: 2,634	2: 34,184
H: 3,271	X: 2,731	n: 12,856	3: 35,293
I: 13,179	Y: 2,359	o: 2,761	4: 33,432
J: 3,962	Z: 2,698	p: 2,401	5: 31,067
K: 2,473	a: 11,196	q: 3,115	6: 34,037
L: 5,390	b: 5,551	r: 15,934	7: 35,796
M: 10,027	c: 11,315	s: 2,698	8: 33,884
N: 9,149	d: 11,421	t: 20,793	9: 33,720

O: 28,680	e: 28,299	u: 2,837	
P: 9,277	f: 2,493	v: 2,854	

Methodology

Through modeling, we will identify and classify each letter to convert hand-written words into computer-typed font. Due to the project constraints, we are limiting the scope to creating a base model that converts handwritten words into digitized characters. At the interest of the stakeholders, this model can be further expanded to convert phrases, and paragraphs. The following three subsections will highlight the various tools, and algorithms used to create this model.

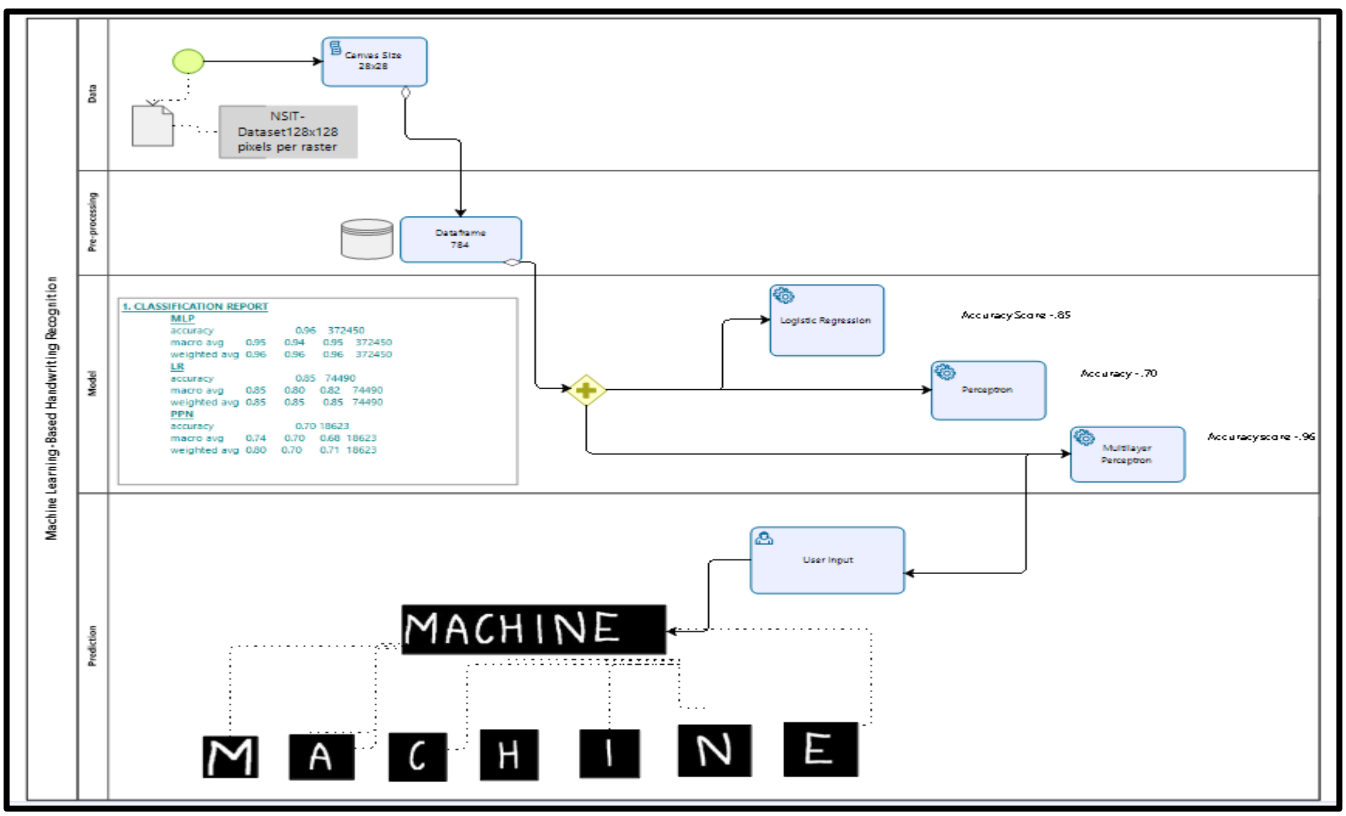


Figure 1: *Bizagi generated Workflow Diagram*

Collaboration Due to the government guidelines regarding social distancing during the pandemic, the group had to work remotely and rely heavily on technology for collaboration. To ensure progress on the project, a combination of WhatsApp group chat, GitHub, and a number of google services such as hangouts, docs, and slides, were used. The WhatsApp groupchat permitted the group to stay in constant communication and platformed scrum meetings while google hangouts enabled the group to conduct more check-ins such through sharing computer screens. Since the work for each deliverable was equally divided up equally amongst each team member, GitHub housed the collaboration regarding the coding effort, and Google docs and Google slides housed the collaborated effort for the report and the presentation, respectively. A high-level breakdown of effort is as follows:

- **Code:** Each member of the team was assigned the construction of a model, and member one was responsible for compiling it
- **Report:** Each member of the team was assigned a section of the report to compose, member two was responsible of compiling it
- **Presentation Slides and Proposal:** Each member of the team was assigned a section of the presentation and proposal and member three is responsible of compiling it

Tools:

To execute this project, we used the **PyCharm IDE** and **Anaconda** libraries to create and host the model. Once we decided our dataset was stable, we scripted the models using Conda libraries in PyCharm. We used logistic regression, perceptron, and multilayer perceptron to classify each handwritten letter and evaluate the model's performance.

Since this is just the base model, we will be using business process modeling notation (BPMN) tool, **Bizagi**, to include detailed and accurate visual representations of all the logic and processes that served as the platform in the development of this application. By clearly outlining our processes, this document can serve as a standard operating procedure for future development of this application.

Preprocessing:

While each character in the original dataset occupies 128x128 pixels per raster (Fig 2a), to avoid heavy computation for this project we decided to first, reduce the size of the image to 56 x 56 pixels (Fig 2b), and then further reduce the canvas size to 28x28 pixels by removing the padding (Fig 2c) and thus creating a 784 feature configuration dataset. Each character is labeled sequentially from "A"- "Z".

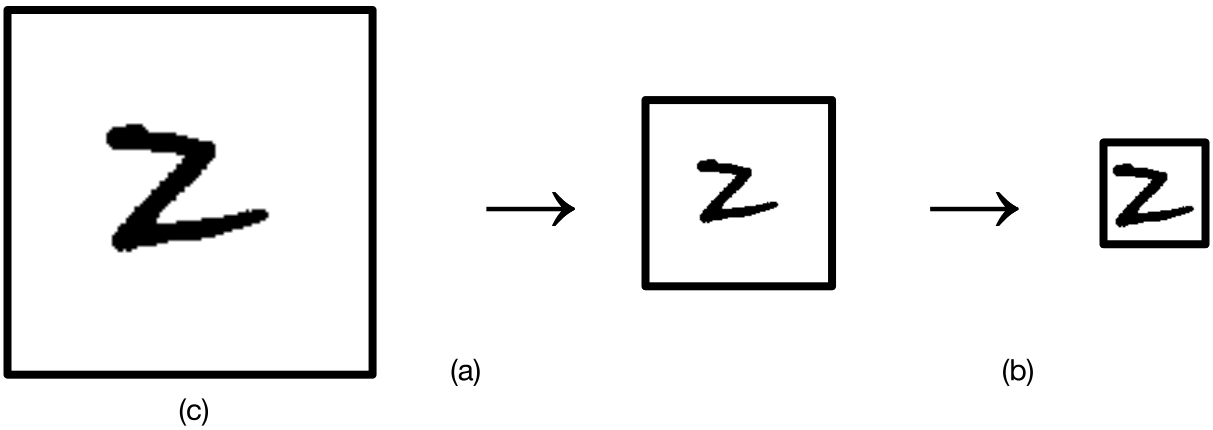


Figure 2: *(a) Original 128 x 128-pixel raster as obtained from the NIST Database, (b) 56 x 56-pixel raster upon resizing the image, (c) 28 x 28-pixel raster upon removing the padding from the resized image*

The package 'tkinter' enables the programming of the user interface. The implementation of this package created a canvas platform where-in the user is able to enter their character or word inputs. The pop-up canvas will convert the image into a NumPy array, traveling column-wise through the canvas and looking for a filled pixel to mark the beginning of a letter. For words, the model continues to traverse and look for a column where there is significant relative blank space to mark the beginning of the second character. Through repeated training, we were able to train the model for precision, meaning the model is intelligent enough to differentiate a break in letters versus the beginning of a second letter. This will be pushed into the model to convert the handwritten scripted character, into a computerized font.

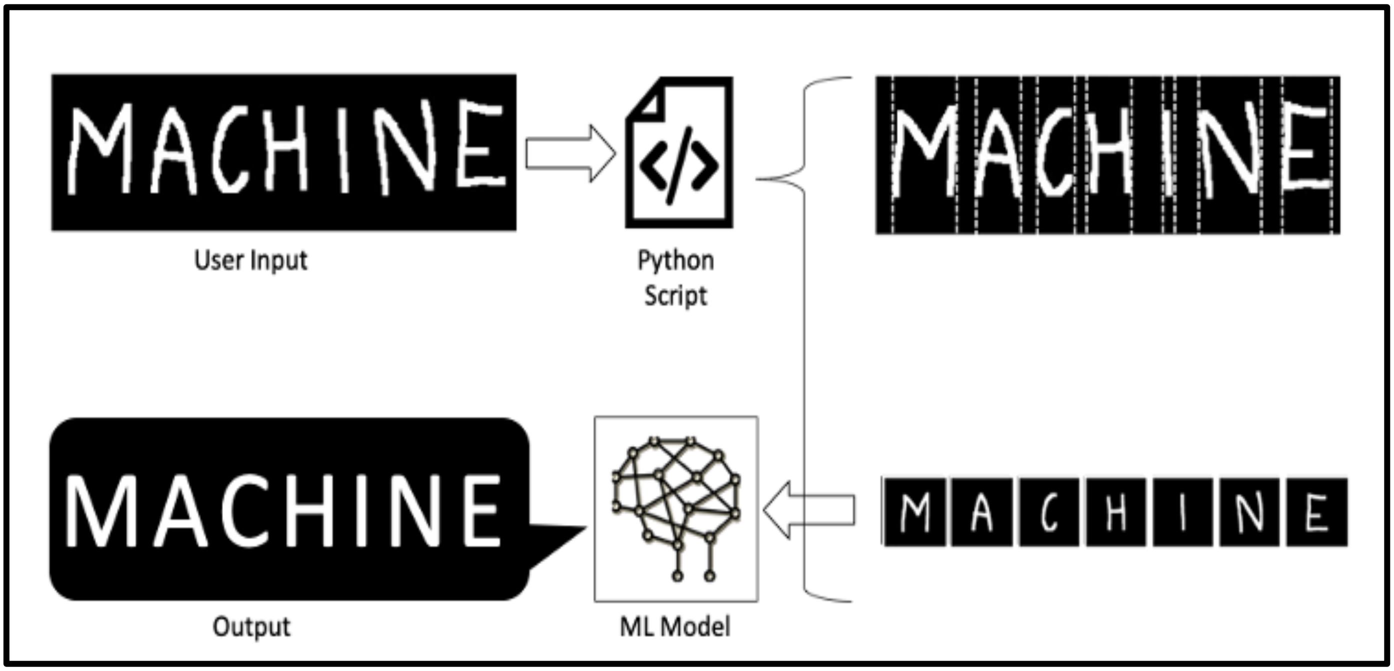


__Figure 2:__ *User- Input Interprets the Model*

__Figure 3__*: User-Interface*

Challenges:

While designing and executing this project we ran into a number of challenges ranging from too large of a dataset to computation capacity. In this section we will highlight our challenges:

#1: The dataset required a heavy computational power to hyper tune the model for the different combinations. Since our computers were unable to accommodate, we overcame this hurdle by breaking the dataset into batches of five characters (i.e. letters A-E and F-I, etc.) and trained and tested the model using these batches.

#2: We faced an imbalance in the number of observations among the characters in the dataset, resulting in complications in testing and training phase (i.e. 10k and 2.5k images for J and K respectively). Our solution to this was creating a script that splits the test and train dataset for each character individually. Once completed, we merged them together to ensure a well-balanced dataset. i

#3: The English alphabet contains letters that appear very similar to each other (i.e. B and P). While training and testing the model, we noticed that the model was struggling to classify these letters. To minimize misclassification for these letters, we trained only these letters with a larger dataset and tuned based on that.

Results

In this section, we will evaluate, discuss and compare the performance of each model.

<u>Logistic Regression</u>

TOTAL: **372450** | MISCLASSIFIED : **10883**				
MODEL: **LOGISTIC REGRESSION** |Over all ACCURACY: **0.85**				
Character:	Attempts:	Correct:	Accuracy in %:	Missclassified with:
A	5	0	0	['K''X''X''Y''X']
B	12	1	8.3	['X''L''L''E''K''A''K''K''K''E''Y']
C	5	2	40	['K''K''K']
D	3	0	0	['Y''Y''Y']
E	13	6	46.2	['Y''L''L''D''Y''K''G']
F	13	3	23.1	['K''P''P''P''T''K''T''P''P''S']
G	3	1	33.3	['S''C']
H	4	2	50	['U''Y']
I	11	3	27.3	['L''S''X''Y''Y''Y''F''Y']
J	12	5	41.7	['P''L''S''S''A''S''K']
K	3	2	66.7	['Y']
L	9	4	44.4	['K''K''K''Y''K']
M	13	0	0	['Y''Y''L''Y''U''K''U''U''K''N''T''K''K']
N	16	4	25	['K''K''U''H''K''Y''K''K''K''Y''K''K']
O	13	6	46.2	['B''D''B''Y''Y''L''J']
P	4	2	50	['T''T']
Q	5	2	40	['O''U''A']
R	13	1	7.7	['K''K''Y''K''K''K''K''T''K''K''K''K']
S	3	2	66.7	['Y']
T	7	1	14.3	['Q''Q''Q''P''Y''P']
U	3	3	100	None
V	10	0	0	['Y''X''K''U''U''U''U''U''U''U']
W	14	9	64.3	['K''L''L''K''K']
X	10	7	70	['L''A''K']
Y	4	4	100	None
Z	4	3	75	['E']

<u>Figure 4:</u> *Logistic Regression Performance Matrix*

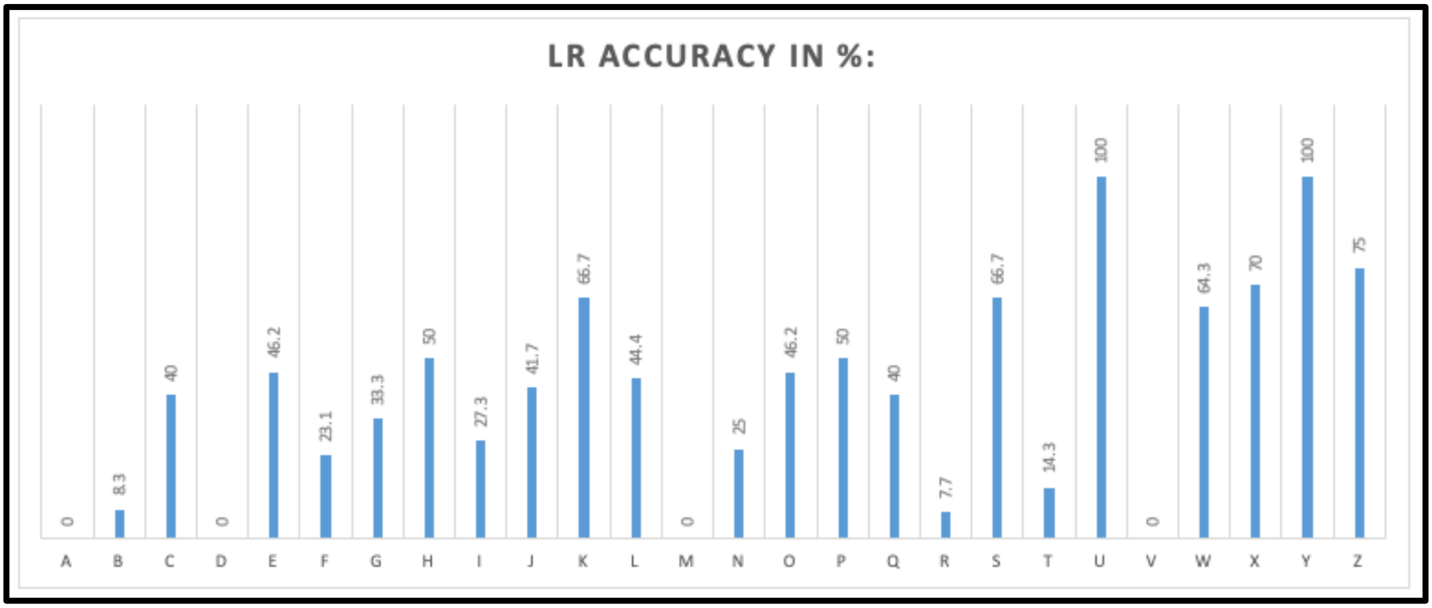


***__Figure 5:__** Logistic Regression Accuracy Chart*

	precision	recall	f1-score	support
A	0.80	0.83	0.82	2739
B	0.75	0.74	0.74	1757
C	0.90	0.89	0.89	4736
D	0.81	0.65	0.72	2022
E	0.78	0.80	0.79	2263
F	0.95	0.80	0.87	226
G	0.84	0.71	0.77	1181
H	0.77	0.60	0.68	1422
I	0.89	0.83	0.86	237
J	0.84	0.62	0.71	1730
K	0.74	0.71	0.73	1084
L	0.93	0.94	0.93	2302
M	0.86	0.87	0.86	2449
N	0.80	0.80	0.80	3874
O	0.86	0.97	0.91	11683
P	0.89	0.90	0.89	3883
Q	0.87	0.67	0.76	1245
R	0.75	0.75	0.75	2321
S	0.90	0.93	0.92	9516
T	0.93	0.94	0.94	4425
U	0.82	0.85	0.84	5796
V	0.89	0.87	0.88	843
X	0.82	0.77	0.79	2135
Y	0.82	0.77	0.80	1241
Z	0.86	0.81	0.83	2130
	0.92	0.81	0.86	1250
accuracy			0.85	74490
macro avg	0.85	0.80	0.82	74490
weighted avg	0.85	0.85	0.85	74490

***__Figure 6:__** Logistic Regression Classification Report*

Logistic regression is a binary classifier. Due to this projects scope, we implemented one versus all extensions to classify all 26 characters. Since we have a large number of features, the learned hypothesis fits the training set well but fails to generalize for the test dataset. There is a high probability that our hypothesis will overfit the training data.

Perceptron

TOTAL: 372450 | MISCLASSIFIED : 5507				
MODEL: PERCEPTRON | Over all ACCURACY: 0.70				
Character:	Attempts:	Correct:	Accuracy in %:	Missclassified with:
A	5	0	0	['K', 'X', 'K', 'Y', 'X']
B	12	0	0	['Y', 'L', 'L', 'H', 'K', 'A', 'Y', 'K', 'Y', 'K', 'D', 'H']
C	5	2	40	['K', 'K', 'K']
D	3	0	0	['H', 'Y', 'Y']
E	13	5	38.5	['Y', 'Y', 'B', 'D', 'D', 'A', 'A', 'A']
F	13	4	30.8	['A', 'P', 'P', 'P', 'T', 'E', 'P', 'P', 'S']
G	3	1	33.3	['C', 'Q']
H	4	0	0	['W', 'A', 'U', 'A']
I	11	1	9.1	['Y', 'J', 'S', 'T', 'Y', 'J', 'Q', 'Y', 'J', 'Y']
J	12	6	50	['R', 'B', 'D', 'Y', 'A', 'Y']
K	3	1	33.3	['B', 'W']
L	9	3	33.3	['Y', 'K', 'K', 'Y', 'A', 'U']
M	13	2	15.4	['Y', 'Y', 'E', 'U', 'E', 'U', 'W', 'K', 'T', 'K', 'A']
N	16	5	31.2	['K', 'K', 'Y', 'A', 'K', 'K', 'Y', 'E', 'X', 'E', 'E']
O	13	0	0	['D', 'D', 'B', 'D', 'D', 'E', 'D', 'C', 'Y', 'D', 'A', 'N', 'J']
P	4	3	75	['Y']
Q	5	3	60	['A', 'A']
R	13	3	23.1	['W', 'J', 'W', 'B', 'X', 'T', 'W', 'F', 'D', 'K']
S	3	1	33.3	['Y', 'A']
T	7	1	14.3	['Q', 'Q', 'S', 'P', 'Y', 'F']
U	3	1	33.3	['B', 'B']
V	10	0	0	['X', 'X', 'A', 'U', 'C', 'B', 'U', 'A', 'B', 'U']
W	14	9	64.3	['K', 'E', 'L', 'U', 'K']
X	10	8	80	['E', 'K']
Y	4	3	75	['E']
Z	4	3	75	['E']

Figure 7: *Perceptron Performance Matrix*

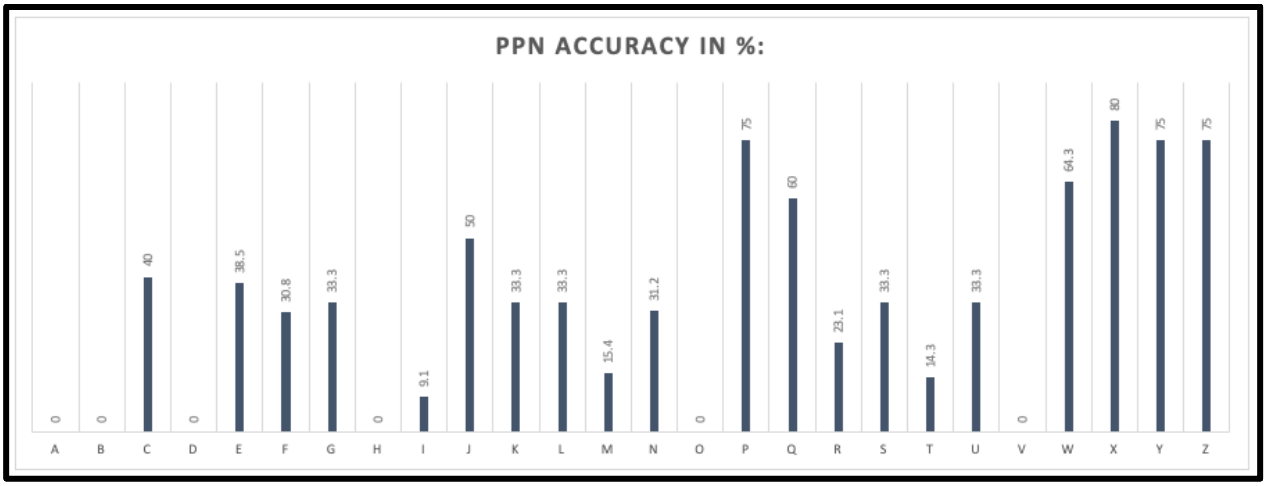


Figure 8: *Perceptron Accuracy Chart*

	precision	recall	f1-score	support
A	0.75	0.73	0.74	691
B	0.45	0.53	0.49	407
C	0.78	0.86	0.82	1163
D	0.25	0.89	0.39	517
E	0.46	0.83	0.59	560
F	0.86	0.78	0.82	54
G	0.89	0.47	0.61	294
H	0.62	0.54	0.57	363
I	0.92	0.66	0.77	68
J	0.73	0.59	0.65	436
K	0.60	0.37	0.46	248
L	0.95	0.76	0.84	616
M	0.69	0.93	0.79	600
N	0.64	0.75	0.69	918
O	0.96	0.39	0.55	2933
P	0.88	0.83	0.86	1024
Q	0.26	0.84	0.40	309
R	0.86	0.25	0.38	579
S	0.92	0.90	0.91	2410
T	0.91	0.89	0.90	1143
U	0.86	0.71	0.78	1417
V	0.87	0.76	0.81	203
X	0.62	0.80	0.70	533
Y	0.68	0.80	0.73	306
Z	0.87	0.65	0.75	533
	0.89	0.71	0.79	298
accuracy			0.70	18623
macro avg	0.74	0.70	0.68	18623
weighted avg	0.80	0.70	0.71	18623

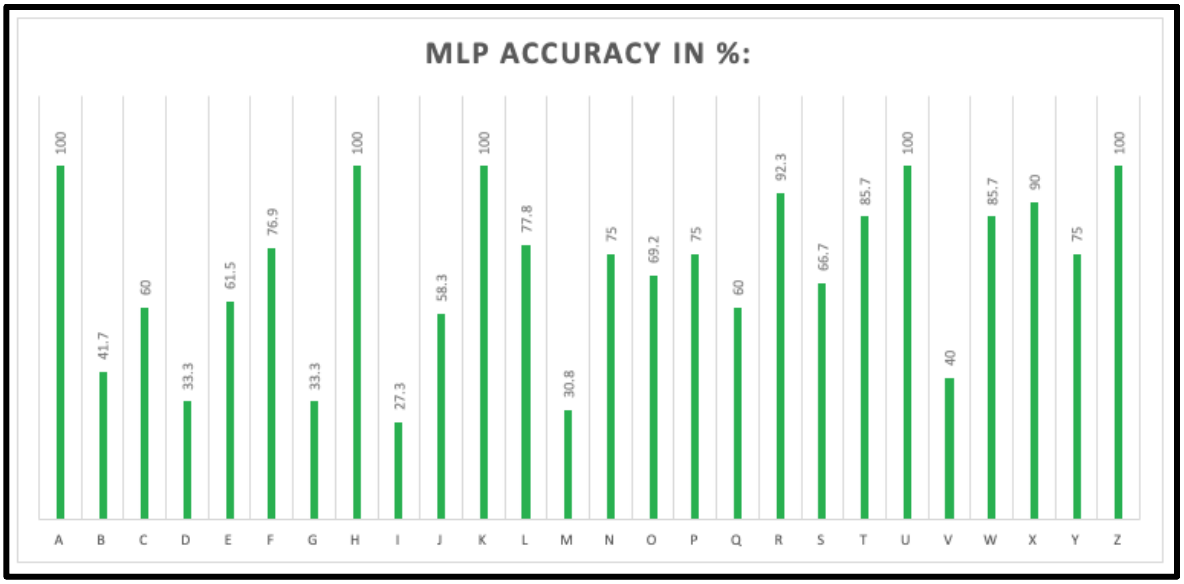
Figure 9: *Perceptron Classification Report*

We continued our model building with basic neural network model i.e., perceptron learner to classify the dataset. The output values of a perceptron can only take one of two values (0 or 1) due to the hard limit function. This was not a good fit because of the accuracy of 70%.

Multilayer perceptron

TOTAL: **372450**| MISCLASSIFIED : **14229**				
MODEL: **MultilayerPerceptron** |Over all ACCURACY: **0.96**				
Character:	Attempts:	Correct:	Accuracy in %:	Missclassified with:
A	5	5	100	None
B	12	5	41.7	['H''H''E''R''K''K''S']
C	5	3	60	['O''L']
D	3	1	33.3	['O''O']
E	13	8	61.5	['K''L''R''C''T']
F	13	10	76.9	['E''E''I']
G	3	1	33.3	['S''C']
H	4	4	100	None
I	11	3	27.3	['T''J''G''T''T''C''T''L']
J	12	7	58.3	['T''L''S''T''X']
K	3	3	100	None
L	9	7	77.8	['K''F']
M	13	4	30.8	['K''K''W''X''K''N''E''R''R']
N	16	12	75	['X''X''K''W']
O	13	9	69.2	['N''N''S''D']
P	4	3	75	['A']
Q	5	3	60	['G''U']
R	13	12	92.3	['L']
S	3	2	66.7	['Y']
T	7	6	85.7	['F']
U	3	3	100	None
V	10	4	40	['U''U''N''U''X''U']
W	14	12	85.7	['U''N']
X	10	9	90	['W']
Y	4	3	75	['X']
Z	4	4	100	None

Figure 10: *MLP Performance Matrix*



<u>Figure 11</u>: *MLP Accuracy Chart*

	precision	recall	f1-score	support
A	0.95	0.96	0.95	13869
B	0.94	0.92	0.93	8668
C	0.96	0.98	0.97	23409
D	0.91	0.90	0.90	10134
E	0.94	0.94	0.94	11440
F	0.96	0.92	0.94	1163
G	0.93	0.92	0.92	5762
H	0.95	0.88	0.91	7218
I	0.94	0.90	0.92	1120
J	0.96	0.92	0.94	8493
K	0.92	0.92	0.92	5603
L	0.97	0.97	0.97	11586
M	0.96	0.96	0.96	12336
N	0.94	0.96	0.95	19010
O	0.97	0.98	0.98	57825
	0.97	0.98	0.98	19341
P	0.92	0.91	0.92	5812
Q	0.92	0.94	0.93	11566
R	0.99	0.98	0.99	48419
S	0.98	0.99	0.98	22495
T	0.97	0.98	0.97	29008
U	0.98	0.98	0.98	4182
V	0.95	0.94	0.95	10784
X	0.94	0.96	0.95	6272
Y	0.97	0.95	0.96	10859
Z	0.96	0.96	0.96	6076
accuracy			0.96	372450
macro avg	0.95	0.94	0.95	372450
weighted avg	0.96	0.96	0.96	372450

<u>Figure 12</u>: *MLP Classification Report*

While the results from logistic regression and perceptron network have been fair, in this project we observed that networks with more than one perceptron spanning across multiple layers can be used to solve complex problems. Therefore, for this classification we used MLP classifier with Adam solver and sigmoid function to achieve significant results.

Summary

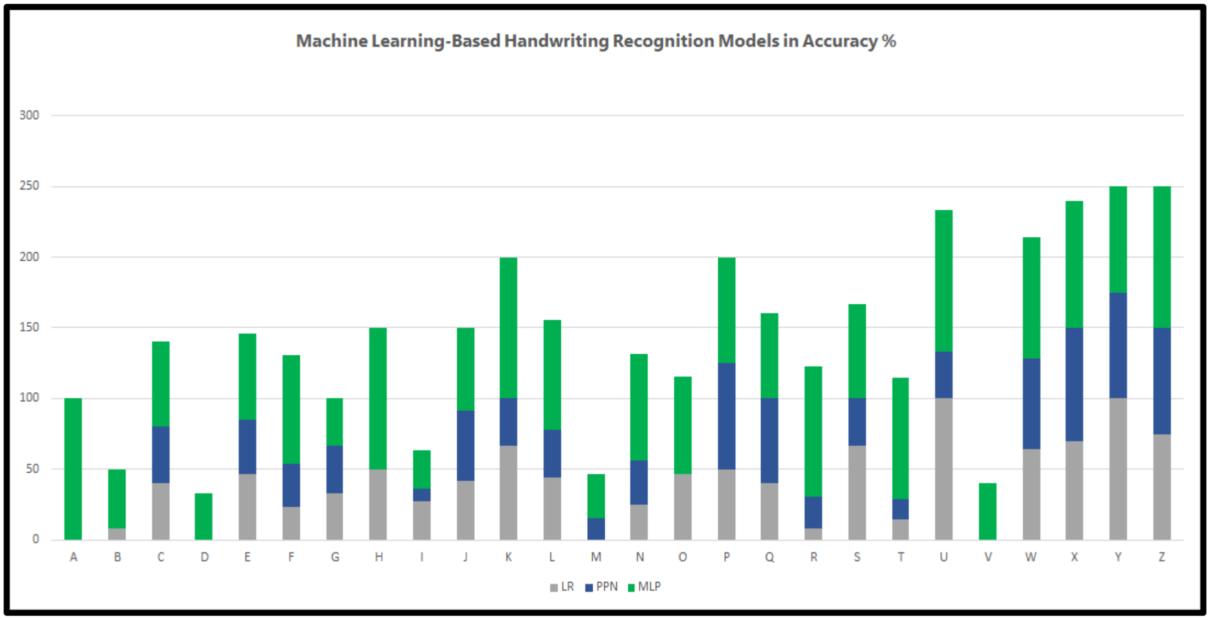


Figure 13: *Accuracy Summary Chart*

This shows the accuracy of all three models. It is evident for letters like A, B, and D, where MLP classifier outperforms compared to other models.

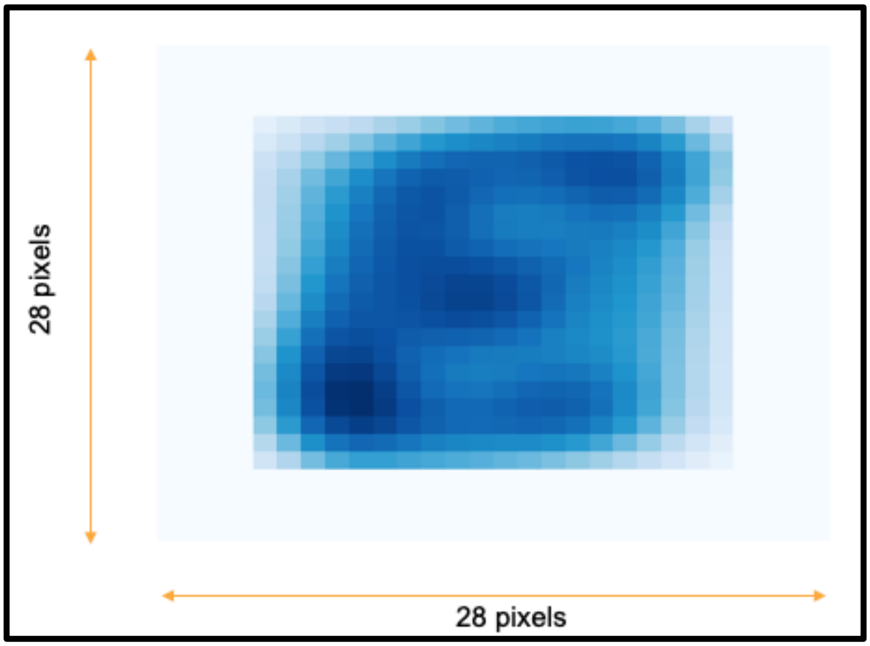


Figure 14: *Pixel Density Graph*

Conclusion

Doctors are well-known for their bad handwriting. Historically, since these notes were only seen by other doctors, this did not pose a problem. However now, doctors are just one component in a wide web of healthcare professionals. To bridge this communication gap, we have developed an application that can translate handwritten notes into a computer-legible format. An advantage of this is that when medical records are passed into the web, it leaves little to no room for miscommunication. Additionally, it decreases stress on the healthcare professional to decrypt the handwriting ultimately resulting in a better healthcare service.

To build this application, we trained a model using a dataset compiled by NIST. To assess the validity and performance, we used logistic regression, perceptron and multi-layer perceptron. In our assessment, we were able to capture a 96% accuracy through the MLP model.

Future Prospects

While this application serves as a base model in bridging the communication gap, there is more work that needs to be done. Currently, the model can decrypt letters and words, but it is capable of processing phrases and paragraphs with proper expansion. Additionally, the UI/ UX can be further developed to be leveraged by a wider user-audience.

References

Center for the Advancement of Health. "Computerized Doctors' Orders Reduce Medication Errors." ScienceDaily. ScienceDaily, 27 June 2007. <www.sciencedaily.com/releases/2007/06/070627084702.htm>.

Grother, Patrick J. “NIST Special Database 19.” NIST, National Institute of Standards and Technology, 27 Apr. 2019, < www.nist.gov/srd/nist-special-database-19 >.