Letter recognition

ITCS-3156 Final Project

Quinn Migliaccio

Introduction

Problem statement

Handwritten character recognition is important for many real world applications. This includes mail sorting, digitizing documents, and text to speech. These are just a few of the uses of handwritten character recognition and there are many more uses out there. Accurate and efficient classification of handwritten letters remains a challenge due to differences in writing such as size, slant, spacing, pressure, form, etc. [5].

Motivation and challenges

A motivation for using ML algorithms to identify handwritten characters so that technology can read handwriting and convert it to text. This is useful for many automated systems such as the post office where letters are scanned and sorted without having to be read by a human. This technology can also be used as assistive technology. It can be used by text to speech to read handwritten messages for those who are visually impaired. A potential concern is that the data used randomly distorted images based on 20 fonts. This could interfere with the recognition of how some people write. The data set also did not contain the images themselves but rather a set of pre decided features. There was also a fairly even distribution of all letters whereas in normal text the distribution would be uneven across letters. Another challenge is that some letters have very similar features such as E and F or O and Q. This can make them hard to distinguish from each other and make it hard for the model.

Summary

The ability to identify handwritten letters is important for many different technologies. This project uses logistic regression and neural networks to classify handwritten letters using UC Irvine's Letter Recognition dataset [4]. Logistic regression is useful for classification tasks because there were more than two possible classes multinomial logistic regression was used. Neural networks are a little more complex than Logistic regression and are able to capture non linear relationships.

Data/Environment

The dataset is called Letter Recognition and comes from UC Irvine Machine Learning Repository. The dataset contains 20,000 samples of English uppercase letters, based on 20 different fonts each with varying distortion. Each sample is represented by 16 numerical features which are:

- **lettr**: capital letter
- **x-box**: Horizontal position of box
- **y-box**: Vertical position of box
- **width**: Width of box
- **high**: Height of box
- **onpix**: Total # on pixels
- **x-bar**: Mean x of on pixels in box
- **y-bar**: Mean y of on pixels in box
- **x2bar**: Mean x variance
- **y2bar**: Mean y variance
- **xybar**: Mean x y correlation
- **x2ybr**: Mean of x * x * y
- **xy2br**: Mean of x * y * y
- **x-ege**: Mean edge count left to right
- **xegvy**: Correlation of x-ege with y
- **y-ege**: Mean edge count bottom to top
- **yegvx**: Correlation of y-ege with x

display(df)

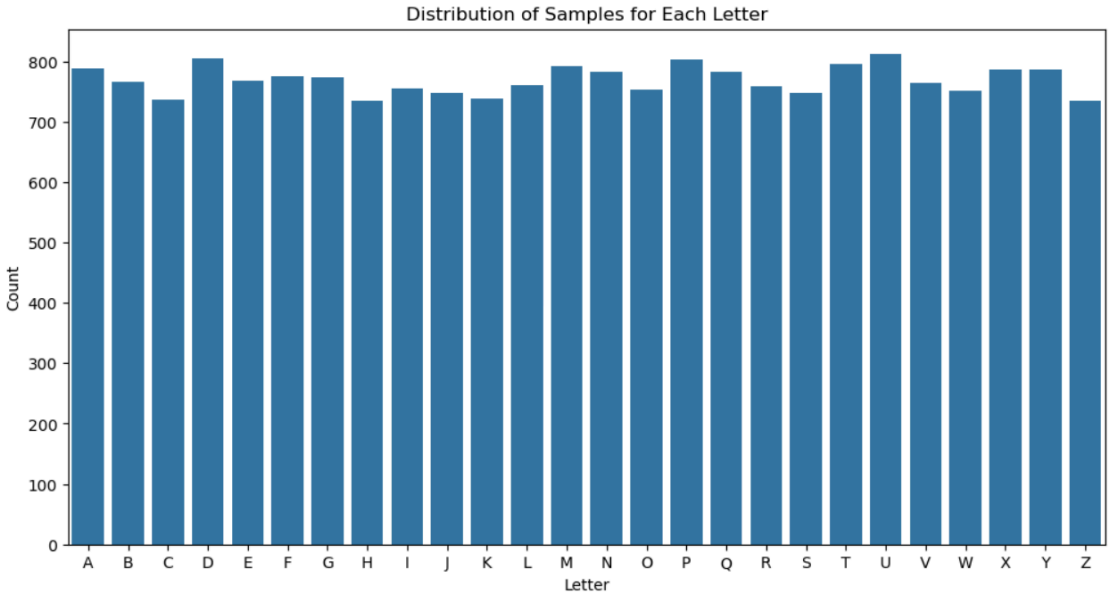
	x-box	y-box	width	high	onpix	x-bar	y-bar	x2bar	y2bar	xybar	x2ybr	xy2br	x-ege	xegvy	y-ege	yegvx	letter
0	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8	T
1	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10	I
2	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9	D
3	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8	N
4	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10	G
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19995	2	2	3	3	2	7	7	7	6	6	6	4	2	8	3	7	D
19996	7	10	8	8	4	4	8	6	9	12	9	13	2	9	3	7	C
19997	6	9	6	7	5	6	11	3	7	11	9	5	2	12	2	4	T
19998	2	3	4	2	1	8	7	2	6	10	6	8	1	9	5	8	S
19999	4	9	6	6	2	9	5	3	1	8	1	8	2	7	2	8	A

20000 rows × 17 columns

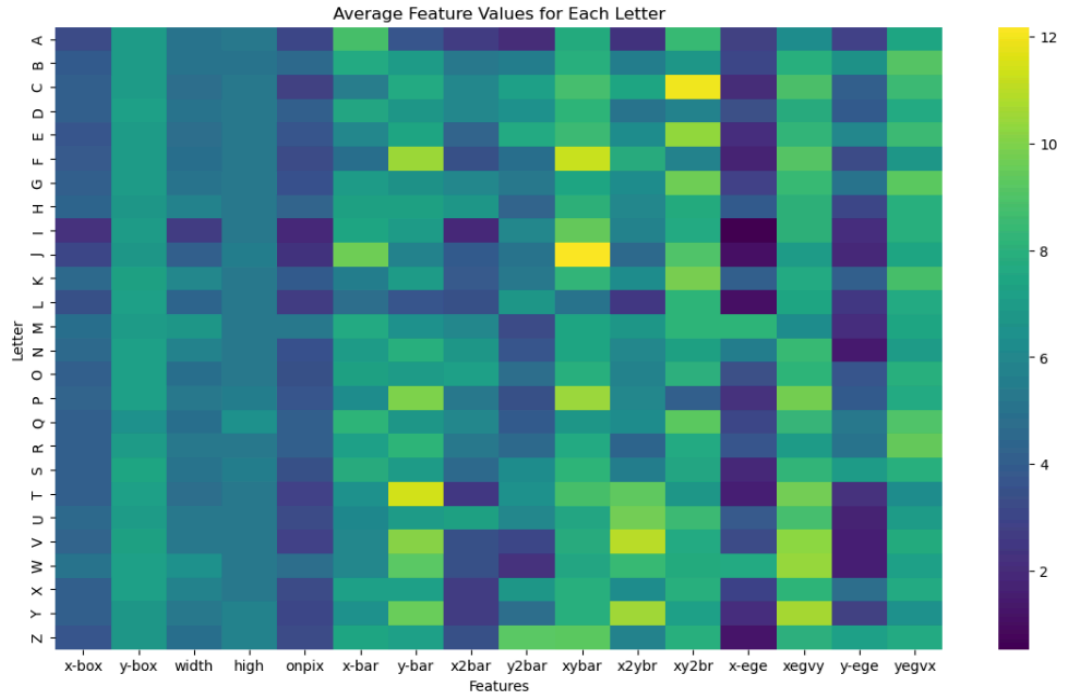
The image shows all 16 features and that there are 20000 rows.

Data visualization & analysis

The plot count shows the distribution of samples for each letter. It shows that there are roughly the same number of samples for each letter. Each letter has between 700 and 800 samples.



The heat map shows the average value of each feature per letter. This was used to see if there were any patterns among the classes. It shows that there is low variability with x-box and y-box but if there is high variability for some of the other features such as x-bar and xy-bar.



Data preprocessing

The Data set was split into training (64%), validation (16%), and test (20%) using sklean's train_test_split. The features were then standardized using sklearn's StandardScalar. Scaling the data improves performance and convergence for both models.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_trn, X_tst, y_trn, y_tst = train_test_split(X, y, train_size=0.8, random_state=42, stratify=y)
X_trn, X_vld, y_trn, y_vld = train_test_split(X_trn, y_trn, train_size=0.8, random_state=42, stratify=y_trn)

# scale
scaler = StandardScaler()
X_trn = scaler.fit_transform(X_trn)
X_vld = scaler.transform(X_vld)
X_tst = scaler.transform(X_tst)
```

Method

logistic regression

Multinomial logistic regression is a slightly modified version of logistic regression that predicts multinomial probability [1]. This means that it was able to target all possible letters (A-Z). Multinomial logistic regression uses the softmax function [2] and solver = lbfgs which supports multinomial loss [1]. To evaluate the model accuracy, precision, recall, and f1-score are looked at. Accuracy is "the proportion of correctly classified instances" [2]. The classification report breaks down the performance for each class [4]. It shows precision, recall, f1-score, and support. Precision is "how many predicted positives are actually positive", reclass is "how many of the actual positives are correctly predicted", f1-score is "the harmonic mean of precision and recall useful for imbalance classes", and support is "the number of occurrences of each class in the dataset" [2].

The confusion matrix is a visual representation of how many times the model correctly or incorrectly predicted a letter. The correct are along the diagonal and the incorrect are everything else. This helps to identify specific letters that the model confuses with each other. Adding the heat map to the confusion matrix just makes it easier to see the accuracy. It makes it so that at a quick glance we can see what stands out.

Neural Network

Multi-layer perceptron (MLP) was used to model nonlinear relationships in the data. They are typically used to model classification tasks and work well with image recognition [3]. The neural network has multiple layers or nodes connected together. The model used has an input layer, two hidden layers one with a size of 100 and the second with 50, and an output layer. The input layer is the layer that receives the features of the dataset. The number of layers in this dataset equals the number of features, so for this dataset 16. The hidden layers are between the input and output layers. The output layer generates the final output based on the data from the hidden layers [3]. Once again accuracy and the classification report were used to see how well the model performed. The loss curve can be used to identify any over or underfitting.

Results

Logistic regression

The first thing done was import the data set. Then the data was split into test, train, and validation sets. After the data was split it was standardized using StandardScaler. The scaled data was used to perform logistic regression. The logistic regression function was imported from sklearn.linear_model. The function was then used to train the model with the hyperparameters solver='lbfgs' and max_iter=1500. The model was then validated and tested. The accuracy and classification report were then printed out in order to analyze the results.

Training Accuracy: 0.782421875
Validation Accuracy: 0.7665625

Classification Report (Validation):

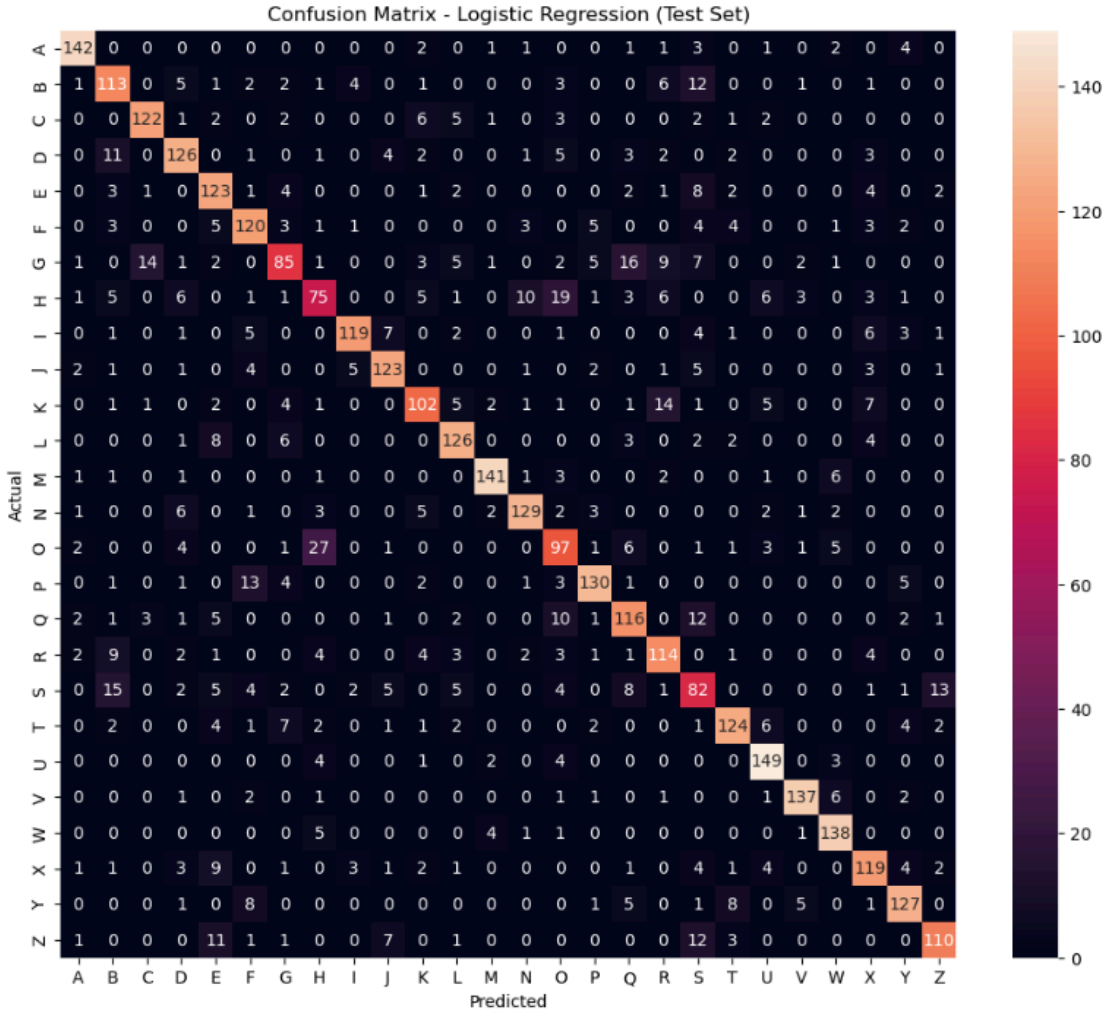
	precision	recall	f1-score	support
A	0.82	0.88	0.85	126
B	0.69	0.82	0.75	123
C	0.80	0.80	0.80	118
D	0.77	0.83	0.80	129
E	0.75	0.79	0.77	123
F	0.78	0.81	0.80	124
G	0.61	0.59	0.60	124
H	0.50	0.44	0.46	117
I	0.89	0.79	0.83	121
J	0.82	0.82	0.82	120
K	0.68	0.62	0.65	118
L	0.83	0.82	0.83	122
M	0.92	0.87	0.89	127
N	0.80	0.80	0.80	125
O	0.64	0.67	0.66	121
P	0.93	0.80	0.86	128
Q	0.73	0.78	0.75	125
R	0.71	0.79	0.75	121
S	0.63	0.43	0.51	120
T	0.79	0.86	0.82	127
U	0.78	0.80	0.79	130
V	0.85	0.87	0.86	122
W	0.85	0.83	0.84	120
X	0.70	0.80	0.75	126
Y	0.83	0.83	0.83	126
Z	0.80	0.76	0.78	117
accuracy			0.77	3200
macro avg	0.77	0.76	0.76	3200
weighted avg	0.77	0.77	0.76	3200

Test Accuracy: 0.77225

Classification Report (Test):

	precision	recall	f1-score	support
A	0.90	0.90	0.90	158
B	0.67	0.74	0.70	153
C	0.87	0.83	0.85	147
D	0.77	0.78	0.78	161
E	0.69	0.80	0.74	154
F	0.73	0.77	0.75	155
G	0.69	0.55	0.61	155
H	0.59	0.51	0.55	147
I	0.89	0.79	0.84	151
J	0.82	0.83	0.82	149
K	0.74	0.69	0.72	148
L	0.79	0.83	0.81	152
M	0.92	0.89	0.90	158
N	0.85	0.82	0.84	157
O	0.60	0.65	0.62	150
P	0.85	0.81	0.83	161
Q	0.69	0.74	0.72	157
R	0.72	0.75	0.74	151
S	0.51	0.55	0.53	150
T	0.83	0.78	0.80	159
U	0.83	0.91	0.87	163
V	0.91	0.90	0.90	153
W	0.84	0.92	0.88	150
X	0.75	0.76	0.75	157
Y	0.82	0.81	0.81	157
Z	0.83	0.75	0.79	147
accuracy			0.77	4000
macro avg	0.77	0.77	0.77	4000
weighted avg	0.77	0.77	0.77	4000

The training accuracy was .78242 validation accuracy was .7665625 and the test accuracy was .77225. This shows that the model performed fairly consistently between the training, validation and test. The accuracy being in the mid to high 70s suggests that there may be nonlinear relationships that linear regression is not able to capture.



Looking at the precision and the matrix we are able to see what letters performed very well and what letters the model struggled with. It shows that the model had the most difficulty correctly identifying G, H, and S. Looking at the precision it shows that .69, .59, and .51 respectively. It also shows that the model was very good at recognizing other letters such as A, M, and U. The matrix also shows which letters were misidentified. We can see that O was misidentified as H 27 times and H was misidentified as O 19 times.

The logistic regression model ended up performing alright but not great. With the accuracy only being in the mid to high 70's that leaves a lot of room for error to occur. Since it appears that there may be non linear relationships that the regression model is unable to predict neural networks can be used since it is able to identify non linear relationships.

Neural network

The data was already imported and split into training, validation, and testing sets. It was also standardized using StandardScaler. MLPClassifier was imported from sklearn.neural_network. MLPClassifier was used to create the structure of the neural network. There were 2 hidden layers the size of the first was 100 and the second was 50. The max_iter was set to 1000 and random_state to 42. The model was then trained, validated, and tested.

Training Accuracy: 0.999609375
Validation Accuracy: 0.948125

Classification Report (Validation):

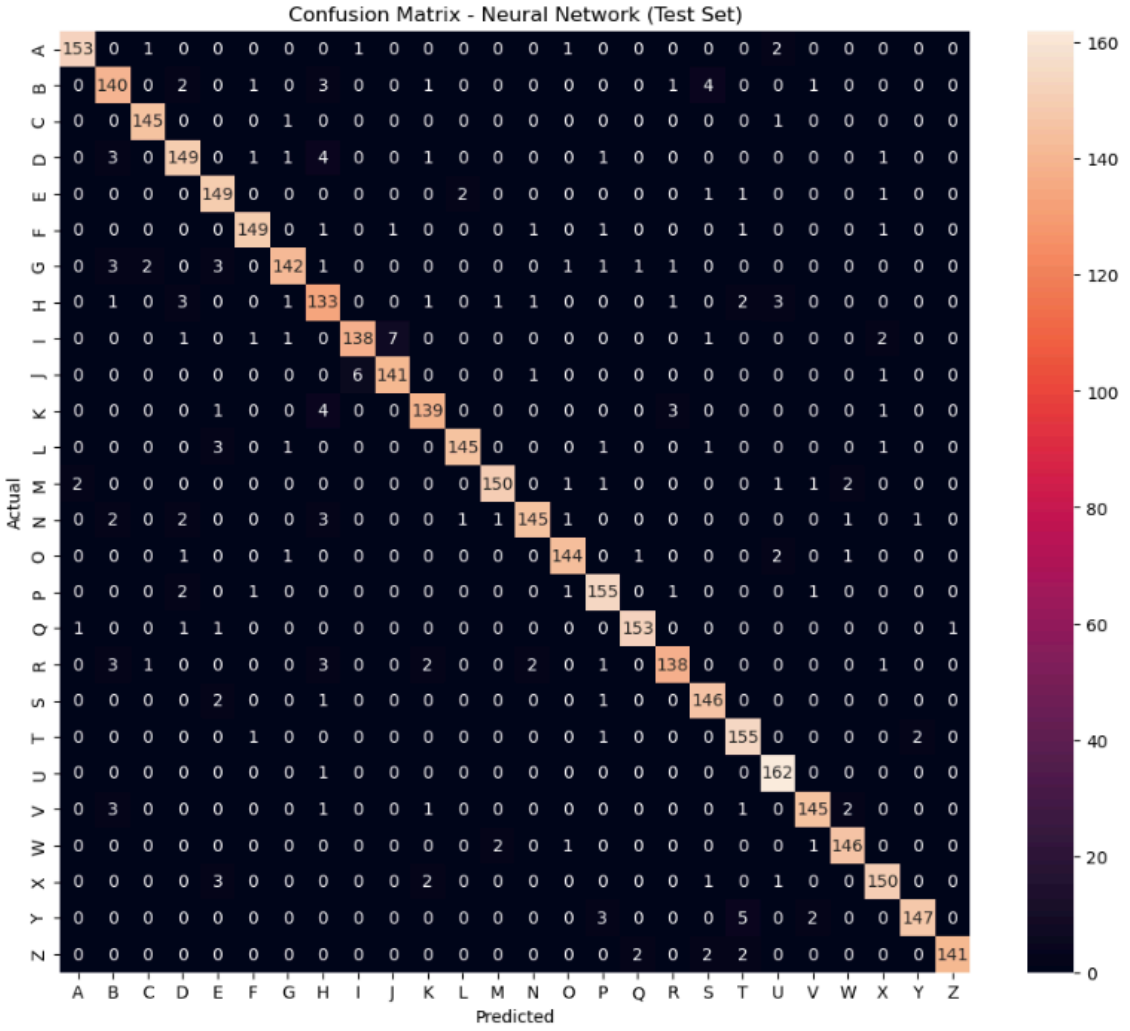
	precision	recall	f1-score	support
A	0.98	0.97	0.98	126
B	0.91	0.95	0.93	123
C	0.95	0.94	0.94	118
D	0.90	0.95	0.92	129
E	0.94	0.96	0.95	123
F	0.94	0.96	0.95	124
G	0.90	0.87	0.89	124
H	0.92	0.93	0.93	117
I	0.96	0.95	0.95	121
J	0.96	0.93	0.94	120
K	0.96	0.94	0.95	118
L	0.97	0.96	0.97	122
M	0.97	0.96	0.96	127
N	0.96	0.91	0.93	125
O	0.91	0.92	0.91	121
P	0.93	0.97	0.95	128
Q	0.93	0.98	0.96	125
R	0.93	0.93	0.93	121
S	0.93	0.94	0.94	120
T	0.98	0.97	0.97	127
U	0.95	0.95	0.95	130
V	0.93	0.94	0.94	122
W	0.99	0.95	0.97	120
X	0.96	0.98	0.97	126
Y	1.00	0.96	0.98	126
Z	0.98	0.97	0.98	117
accuracy			0.95	3200
macro avg	0.95	0.95	0.95	3200
weighted avg	0.95	0.95	0.95	3200

Test Accuracy: 0.95

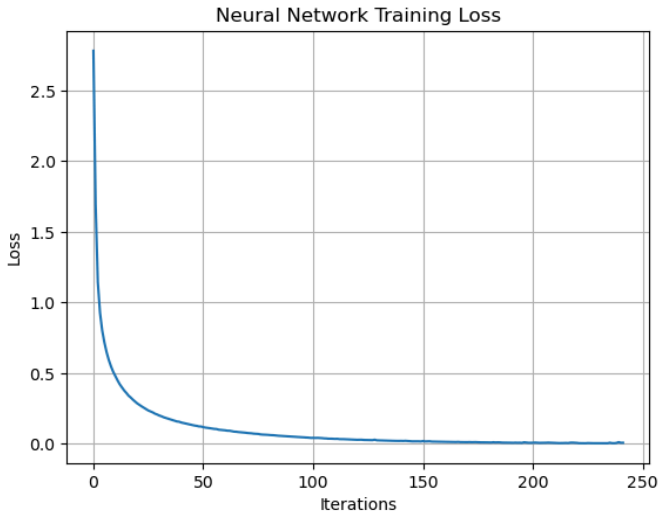
Classification Report (Test):

	precision	recall	f1-score	support
A	0.98	0.97	0.97	158
B	0.90	0.92	0.91	153
C	0.97	0.99	0.98	147
D	0.93	0.93	0.93	161
E	0.92	0.97	0.94	154
F	0.97	0.96	0.96	155
G	0.96	0.92	0.94	155
H	0.86	0.90	0.88	147
I	0.95	0.91	0.93	151
J	0.95	0.95	0.95	149
K	0.95	0.94	0.94	148
L	0.98	0.95	0.97	152
M	0.97	0.95	0.96	158
N	0.97	0.92	0.94	157
O	0.96	0.96	0.96	150
P	0.93	0.96	0.95	161
Q	0.97	0.97	0.97	157
R	0.95	0.91	0.93	151
S	0.94	0.97	0.95	150
T	0.93	0.97	0.95	159
U	0.94	0.99	0.97	163
V	0.96	0.95	0.95	153
W	0.96	0.97	0.97	150
X	0.94	0.96	0.95	157
Y	0.98	0.94	0.96	157
Z	0.99	0.96	0.98	147
accuracy			0.95	4000
macro avg	0.95	0.95	0.95	4000
weighted avg	0.95	0.95	0.95	4000

The training accuracy was .9996 which is very high. The validation accuracy was .948125 and the test accuracy was .95. This means that this model performed very well. The precision, recall, and f1-scores were high for all classes.



The confusion matrix for the neural network also looks very good. The colors along the diagonal are light and the numbers are high. This means that most letters were correctly identified. The matrix does show a few misidentifications but not very many. The largest number of misclassifications were between I and J. I was misclassified as J 7 times and J was misclassified as I 6 times. This result was similar to what was seen in the confusion matrix for logistic regression but this model had much fewer errors in all other identifications.



The loss curve shows that there was a quick initial drop meaning that error was reduced and the patterns were successful early on. The curve then continuous showing that it converges at 0 losses around 200 iterations.

The neural network significantly outperformed the logistic regression. This is due to neural networks being able to identify non linear relationships. The logistic regression confused letters with similar features but the neural network was much better with identifying the letters and had far fewer errors. The low accuracy of the logistic regression and high accuracy of the neural network suggest that neural network was the better predictor for this dataset.

Conclusion

This project used logistic regression and neural network to classify handwritten capital letters. The neural network had a test accuracy of 95% and logistic regression had a test accuracy of 77%. This means that the neural network outperformed the logistic regression suggesting that neural networks work better with recognition tasks.

One of my main challenges was deciding what ML models to use. I decided on starting with logistic regression since it is simple and works well with classification problems. It helped me understand the data a little better. Since logistic regression is linear and it was not performing great I decided to use neural network work as the second model. The neural network was able to capture the nonlinear relationship and was able to perform very well. This taught me more about the importance of model selection.

References & Acknowledgements

The following resources were used to gather general information about logistic regression and neural networks along with how to use them using built in libraries. I also reviewed homework material and other material from the canvas course to complete this project.

[1] Brownlee, J. (2020, December 31). *Multinomial logistic regression with python*. MachineLearningMastery. https://machinelearningmastery.com/multinomial-logistic-regression-with-python/

[2] GeeksforGeeks. (2025a, February 3). *Logistic regression in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/understanding-logistic-regression/

[3] GeeksforGeeks. (2025b, March 10). *Classification using Sklearn Multi-layer Perceptron*. GeeksforGeeks. https://www.geeksforgeeks.org/classification-using-sklearn-multi-layer-perceptron/

[4] Slate, D. (1990, December 31). Letter recognition. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/59/letter+recognition

[5] Smith, T. heodora L. (1954). *Six basic factors in handwriting classification*. Journal of Criminal Law and Criminology. https://scholarlycommons.law.northwestern.edu/cgi/viewcontent.cgi?article=4211&context=jclc

Code

https://github.com/Quinnsadilla/Letter-Recognition.git