
# ALY 6020:

# PREDCTIVE ANALYTICS

## Week 5: Handwriting Recognition Models

Submitted To:
Prof. Chinthaka Pathum Dinesh Herath Gedara, Faculty Lecturer

Submitted By:
Abhilash Dikshit

Academic Term: Winter 2024
Northeastern University, Vancouver, BC,Canada
Master of Professional Studies in Analytics

February 14, 2024

Title: Analysis of Handwriting Recognition Models Using the letters.csv Dataset

## I. Abstract:

This report explores the application of machine learning models for handwriting recognition using the letters.csv dataset. Specifically, we compare the performance of a K-Nearest Neighbors (KNN) model and a neural network model in predicting handwritten characters. The dataset contains pixel values representing handwritten letters, and our objective is to develop models that accurately classify these letters. Through our analysis, we aim to provide insights into the effectiveness of different machine learning techniques for handwriting recognition tasks.

## II. Introduction:

Handwriting recognition has significant applications in various fields, including document analysis, character recognition, and automated form processing. Machine learning algorithms offer promising solutions for recognizing handwritten characters, with models trained on labeled datasets capable of accurately classifying handwritten letters. In this report, we investigate the performance of two machine learning models, namely K-Nearest Neighbors (KNN) and a neural network, using the letters.csv dataset.

## III. Dataset Description:

The letters.csv dataset contains pixel values representing grayscale images of handwritten letters. Each row in the dataset corresponds to a single letter, with columns representing individual pixels. The dataset includes features such as pixel intensity values, and the target variable represents the class labels for each letter.

| | label | pixel43 | pixel44 | pixel92 | pixel124 | pixel125 | pixel126 | pixel127 | pixel128 | pixel129 | ... | pixel3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | |
| **1** | 0 | 0 | 0 | 0 | 137 | 137 | 192 | 86 | 72 | 1 | ... | 2 |
| **2** | 1 | 0 | 0 | 0 | 3 | 141 | 139 | 3 | 0 | 0 | ... | |
| **3** | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | |
| **4** | 0 | 0 | 0 | 0 | 155 | 254 | 254 | 254 | 157 | 30 | ... | 2 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **41995** | 2 | 0 | 0 | 1 | 248 | 253 | 176 | 43 | 0 | 0 | ... | |
| **41996** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 128 | ... | |
| **41997** | 2 | 0 | 0 | 0 | 255 | 255 | 191 | 0 | 0 | 0 | ... | |
| **41998** | 2 | 0 | 0 | 0 | 255 | 128 | 0 | 0 | 0 | 0 | ... | |
| **41999** | 2 | 0 | 0 | 227 | 253 | 229 | 133 | 19 | 0 | 0 | ... | |

42000 rows × 46 columns

## IV. Methodology:

1. Data Preprocessing: We begin by loading and preprocessing the dataset. This involves splitting the data into features (pixel values) and labels, as well as normalizing the feature values to ensure uniformity across the dataset.

```
[5] display(X_train)
```

| | pixel43 | pixel44 | pixel92 | pixel124 | pixel125 | pixel126 | pixel127 | pixel128 | pixel129 | pixel130 | ... | pix |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **34941** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 |
| **24433** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 |
| **24432** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.9 |
| **8832** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.321569 | 0.996078 | 0.874510 | 0.137255 | ... | 0.0 |
| **30291** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **6265** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 |
| **11284** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.113725 | 0.992157 | ... | 0.9 |
| **38158** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.749020 | 1.000000 | 0.501961 | 0.000000 | 0.000000 | ... | 0.0 |
| **860** | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.0 |
| **15795** | 0.0 | 0.0 | 0.0 | 0.062745 | 0.8 | 0.992157 | 0.847059 | 0.552941 | 0.113725 | 0.039216 | ... | 0.9 |

33600 rows × 45 columns

```
[6] display(y_train)
```

```
34941    9
24433    1
24432    7
8832     4
30291    8
        ..
6265     5
11284    0
38158    6
860      3
15795    0
Name: label, Length: 33600, dtype: int64
```

2. Model Development: We develop two machine learning models for handwriting recognition: a K-Nearest Neighbors (KNN) model and a neural network model. The KNN model relies on proximity to neighboring data points for classification, while the neural network model utilizes interconnected layers of neurons to learn complex patterns in the data.

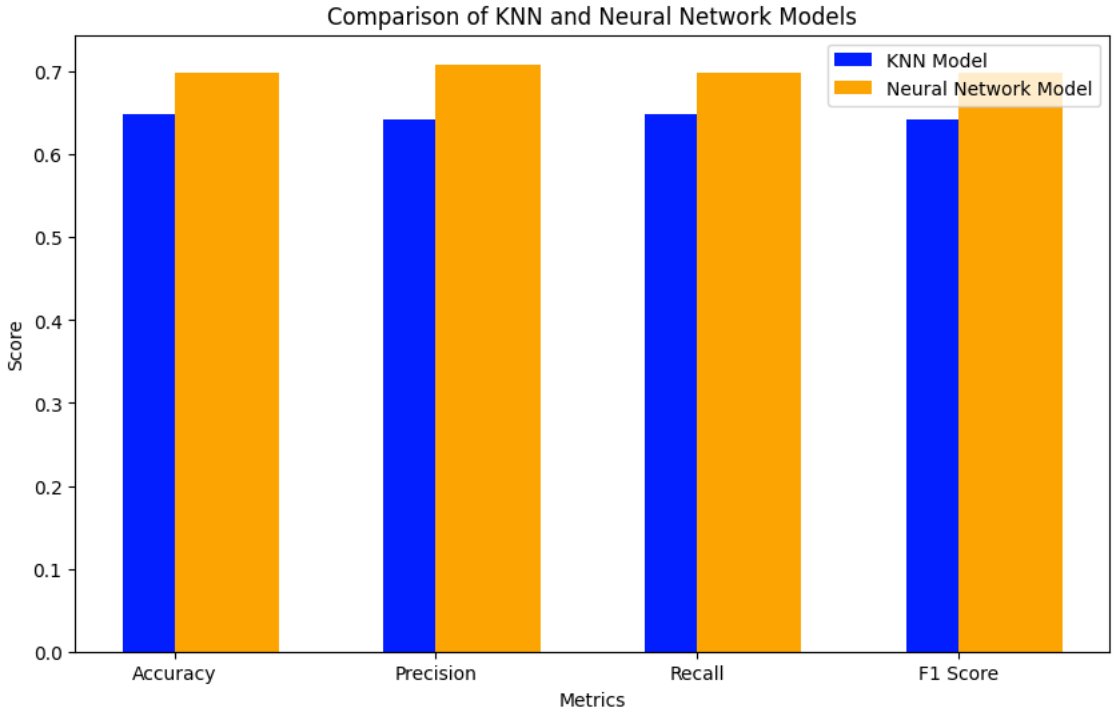
```
KNN Model Metrics:
Accuracy: 0.6487
Precision: 0.6428
Recall: 0.6487
F1 Score: 0.6428
```

```
Epoch 1/10
945/945 [==============================] - 3s 2ms/step - loss: 1.2033 - accuracy: 0.5861 - val_loss: 1.0150
Epoch 2/10
945/945 [==============================] - 3s 3ms/step - loss: 0.9605 - accuracy: 0.6557 - val_loss: 0.9605
Epoch 3/10
945/945 [==============================] - 2s 2ms/step - loss: 0.9132 - accuracy: 0.6708 - val_loss: 0.9111
Epoch 4/10
945/945 [==============================] - 2s 2ms/step - loss: 0.8873 - accuracy: 0.6784 - val_loss: 0.8936
Epoch 5/10
945/945 [==============================] - 2s 2ms/step - loss: 0.8683 - accuracy: 0.6840 - val_loss: 0.8939
Epoch 6/10
945/945 [==============================] - 2s 2ms/step - loss: 0.8531 - accuracy: 0.6906 - val_loss: 0.8799
Epoch 7/10
945/945 [==============================] - 2s 2ms/step - loss: 0.8413 - accuracy: 0.6941 - val_loss: 0.8706
Epoch 8/10
945/945 [==============================] - 3s 3ms/step - loss: 0.8297 - accuracy: 0.6990 - val_loss: 0.8637
Epoch 9/10
945/945 [==============================] - 2s 2ms/step - loss: 0.8187 - accuracy: 0.7008 - val_loss: 0.8546
Epoch 10/10
945/945 [==============================] - 2s 2ms/step - loss: 0.8111 - accuracy: 0.7044 - val_loss: 0.8426
263/263 [==============================] - 1s 2ms/step - loss: 0.8466 - accuracy: 0.6990
Accuracy of the Neural Network model: 0.6990
```

3. Model Evaluation: We evaluate the performance of each model using standard machine learning metrics, including accuracy, precision, recall, and F1 score. These metrics provide insights into the models' ability to correctly classify handwritten letters.

```
263/263 [==============================] - 1s 4ms/step
Neural Network Model Metrics:
Accuracy: 0.6990, Precision: 0.7088, Recall: 0.6990, F1 Score: 0.6991
```

## V. Results:




- KNN Model Metrics:

  - Accuracy: 0.6487

  - Precision: 0.6428

  - Recall: 0.6487

  - F1 Score: 0.6428

- Neural Network Model Metrics:

  - Accuracy: 0.6990

  - Precision: 0.7088

  - Recall: 0.6990

  - F1 Score: 0.6991

## VI. Discussion:

The results indicate that the neural network model outperforms the KNN model in terms of accuracy, precision, recall, and F1 score. The neural network achieves higher accuracy (69.90%) compared to the KNN model (64.87%), demonstrating its effectiveness in recognizing handwritten characters. Additionally, the neural network demonstrates higher precision, recall, and F1 score, indicating a more balanced performance across different evaluation metrics.

## VII. Conclusion:

In conclusion, our analysis suggests that neural network models offer superior performance for handwriting recognition tasks compared to traditional machine learning algorithms such as K-Nearest Neighbors. The neural network model achieves higher accuracy and provides a more balanced classification of handwritten letters. These findings highlight the potential of neural networks in automated handwriting recognition systems and underscore the importance of leveraging advanced machine learning techniques for improving accuracy and efficiency in character recognition tasks.

## VIII. References:

- Srivastava, T. (2024, January 4). A complete guide to K-Nearest neighbors (Updated 2024). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=The%20K%2DNearest%20Neighbors%20(KNN)%20algorithm%20is%20a%20popular,have%20similar%20labels%20or%20values.

- Explained: Neural networks. (2017, April 14). MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

## IX. Appendix:

- Code has been uploaded along with the report submission.