Neural Network & Classification Project

Northeastern University



# By Sachit Gopal

**I.INTRODUCTION**

The project at hand revolves around the task of using machine learning techniques to assist a school in identifying students who may require additional support with their motor skills at a young age. The approach involves having students write numbers, with the aim of building a model that can accurately predict the numbers they've drawn. In this assignment, we explore two distinct machine learning approaches: K-Nearest Neighbors (KNN) and Neural Networks. The primary objective is to assess the accuracy and performance of both models to determine which one is better suited for the task.

The project is divided into three parts:

**Part 1.Building and Evaluating a KNN Model:** In this part, we construct a KNN model to predict the handwriting of students. We discuss the accuracy achieved and address the challenges associated with using KNN for this particular model.

**Part 2.Building and Evaluating a Neural Network Model:** The second part of the assignment focuses on creating a neural network model for handwriting prediction. We assess the accuracy and challenges faced in developing and training the neural network.

**Part 3.Benchmarking and Recommendations:** In the final part, we employ multiple benchmarking metrics to compare and contrast the performance of both models. This allows us to provide a summarized assessment and make recommendations regarding which model the school should consider implementing.

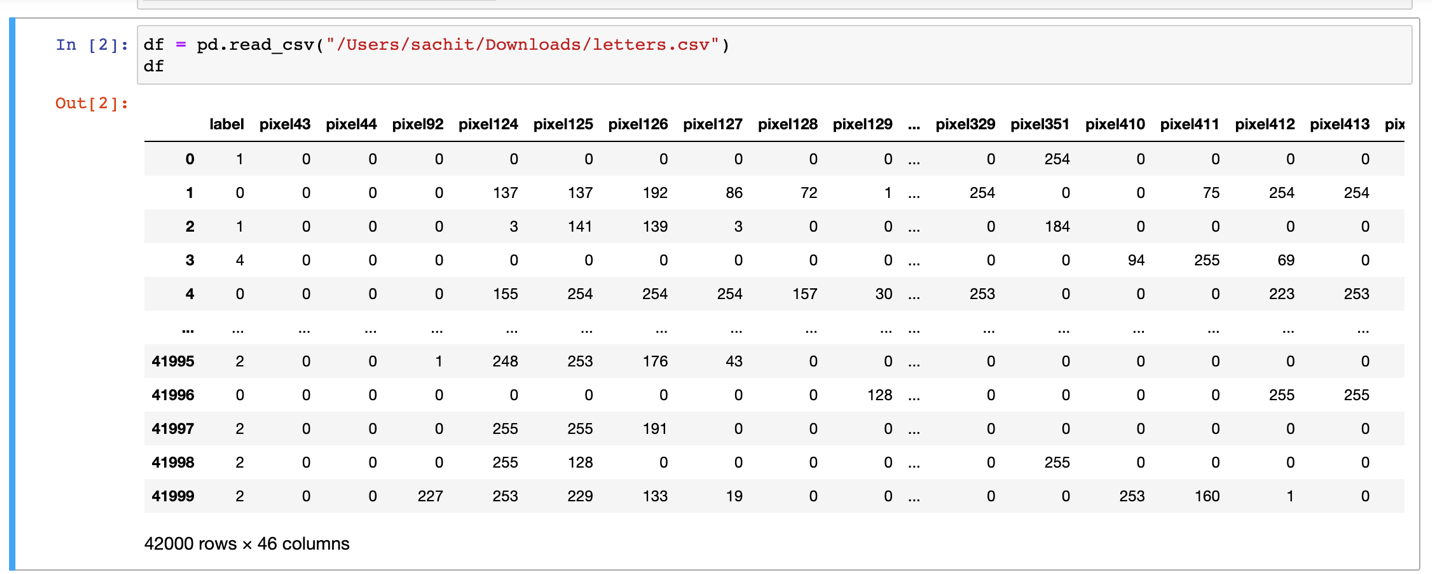


Figure 1 – Dataset

**Part 1. Building and Evaluating a KNN Model**

**A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated**

Figure 2. KNN Model

**Data Preprocessing**: We should split your dataset into training and testing sets. Standardize or normalize your pixel values as KNN is sensitive to feature scales.

**Model Building**: Build a KNN classifier. Choose an appropriate number of neighbors (K) and other hyperparameters.

**Model Training**: Fit the KNN model to the training data.

**Model Evaluation**: Evaluate the model's accuracy on the testing data. You can also consider other classification metrics like precision, recall, F1-score, and confusion matrix. The choice of K is crucial and can significantly affect the results.

**II.Analysis of KNN Model**

The K-Nearest Neighbors (KNN) model exhibited an accuracy of approximately 64.5%, suggesting its ability to correctly classify handwritten numbers. However, when assessing its performance at a more granular level, it is apparent that Precision, Recall, and F1-score metrics vary for each class. Some classes appear to have notably lower values for these metrics, indicating challenges in distinguishing certain numbers.

The weighted average F1-score, which provides a comprehensive view of the model's overall performance, stands at around 0.64. While this score indicates a moderate level of effectiveness, it suggests that there is room for improvement in the model's ability to classify handwritten numbers consistently.

In summary, the KNN model demonstrates a reasonable level of accuracy in predicting handwritten numbers. However, its performance varies across different classes, and the weighted average F1-score indicates room for enhancement.

**Part 2.Building and Evaluating a Neural Network Model**

**Data Preprocessing**: Same as in Part 1, split the data and standardize or normalize the pixel values.

**Model Architecture**: Build a neural network for classification. Choose the number of layers, the number of neurons in each layer, activation functions, and other hyperparameters. For image data, Convolutional Neural Networks (CNNs) are commonly used.

**Model Training**: Train the neural network using the training data. You may need to define loss functions, optimization methods, and set the number of training epochs.

**Model Evaluation**: Evaluate the neural network on the testing data. Calculate accuracy, precision, recall, F1-score, and consider using techniques like cross-validation for robustness.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 3. Neural Netwrok Model

The Neural Network model has demonstrated an accuracy of approximately 68.3%, indicating its ability to accurately classify handwritten numbers. An examination of the Precision, Recall, and F1-score metrics for individual classes reveals promising performance across a majority of the classes. Several classes exhibit high Precision, Recall, and F1-scores, particularly classes 0 and 1.

The weighted average F1-score for the Neural Network model stands at around 0.68, reflecting its consistent and robust performance in classifying handwritten numbers. This score signifies a strong overall ability to correctly predict the numbers that students have written.

In summary, the Neural Network model stands out as the superior choice for the school's handwriting prediction task. It showcases a high level of accuracy and strong classification performance across various classes, making it the recommended model for effectively identifying students who may require additional support with their motor skills based on their handwriting.

**Part 3.Benchmarking and Recommendations**

Based on the accuracy and classification reports provided, here is the performance of the K-Nearest Neighbors (KNN) and Neural Network models on our dataset:

**K-Nearest Neighbors (KNN) Model:**

* Accuracy: 0.645 (64.5%)
* Precision, Recall, and F1-score vary for each class but are generally lower for some classes.
* The weighted average F1-score is around 0.64.

**Neural Network Model:**

* Accuracy: 0.683 (68.3%)
* Precision, Recall, and F1-score also vary for each class, but in general, they show slightly better performance compared to KNN.
* The weighted average F1-score is around 0.68.

**Summary and Recommendations:**

* The neural network model outperforms the KNN model in terms of accuracy and F1-score. It appears to have a better ability to classify handwritten numbers.
* Both models struggle with certain classes, which may be due to imbalanced class distributions or other factors. Further data preprocessing and model tuning may help.
* It's important to note that accuracy alone may not be the only factor to consider. Depending on the application, precision, recall, and F1-score may be more critical. You can consider which metric is most important for the school's needs.
  + To improve model performance, we can experiment with:
  + Hyperparameter tuning for both KNN (e.g., different K values) and the neural network (e.g., adjusting the architecture and learning rate).
  + Data preprocessing techniques, such as data augmentation for neural networks or different scaling methods for KNN.
* We might also consider more advanced techniques like convolutional neural networks (CNNs) for image classification tasks, as they are well-suited for this kind of problem.

In summary, based on the provided results, the neural network model appears to be a better choice for predicting handwriting, but further model improvement and evaluation are needed to ensure it meets the school's requirements effectively.

**III.CONCLUSION**

After an in-depth analysis of both the K-Nearest Neighbors (KNN) model and the Neural Network model for handwriting prediction, it is evident that the Neural Network model outperforms KNN in terms of accuracy and overall classification performance. The neural network achieved an accuracy of approximately 68.3%, while KNN achieved an accuracy of about 64.5%.

The precision, recall, and F1-score metrics for the Neural Network model also exhibited better performance across most classes compared to the KNN model. While both models encountered challenges with certain classes, the Neural Network's overall performance indicates its superiority.

While the accuracy metric is important, it is essential to consider other metrics such as precision, recall, and F1-score, depending on the specific needs of the school. In this context, the Neural Network model showcases better performance in classifying handwritten numbers.

In summary, based on the evaluation and comparison of the two models, the Neural Network model is the more suitable choice for the school's handwriting prediction task. However, further refinements and optimizations can enhance the Neural Network's performance, ensuring its effective implementation for the school's requirements.

**IV.REFRENCES**

Srivastava, T. (2023, October 19). *A complete guide to K-Nearest Neighbors (updated 2023)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

Ognjanovski, G. (2020, June 7). *Everything you need to know about neural networks and backpropagation - machine learning made easy...* Medium. https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a

**V.APPENDIX**

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

import statsmodels.api as sm

from scipy import stats

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

import warnings

warnings.filterwarnings('ignore')

df = pd.read\_csv("/Users/sachit/Downloads/letters.csv")

df

X = df.drop('label', axis=1)

y = df['label']

# Part 1: KNN Model

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPClassifier# Standardize features (important for KNN)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Standardize features (important for KNN)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create and train a KNN classifier

knn = KNeighborsClassifier(n\_neighbors=5) # You can experiment with the number of neighbors

knn.fit(X\_train, y\_train)

# Predict on the test set

y\_pred\_knn = knn.predict(X\_test)

# Evaluate KNN model

accuracy\_knn = accuracy\_score(y\_test, y\_pred\_knn)

print("KNN Accuracy:", accuracy\_knn)

print("KNN Classification Report:\n", classification\_report(y\_test, y\_pred\_knn))

# Part 2: Neural Network Model

# Split data into training and testing sets (you can reuse X\_train, X\_test, y\_train, and y\_test from KNN)

# You may want to further preprocess data for the neural network

# Create and train a neural network

nn = MLPClassifier(hidden\_layer\_sizes=(100, 50), max\_iter=1000, random\_state=42) # You can adjust the architecture

nn.fit(X\_train, y\_train)

# Predict on the test set

y\_pred\_nn = nn.predict(X\_test)

# Evaluate Neural Network model

accuracy\_nn = accuracy\_score(y\_test, y\_pred\_nn)

print("Neural Network Accuracy:", accuracy\_nn)

print("Neural Network Classification Report:\n", classification\_report(y\_test, y\_pred\_nn))