Text Classification

ALY6020: Predictive Analytics

Module 5 Assignment

Submitted by:

Naina Gupta

Submitted on:

6/25/2023

Introduction

This report is a summary of models built to see if we can predict the numbers drawn by students in a writing test. To accomplish this, we will use a dataset of pixel intensity values of the letters and build a simplistic KNN model while starting the research. Subsequently, we will compare the results with something more complicated like neural networks. The goal of this project is to decide if the writing test can be used to understand which students may need to work on their motor skills at a young age.

Jupyter Notebook is leveraged to conduct a comprehensive examination of the dataset. Our methodology constitutes extraction of valuable insights from the data by building three types of models, evaluating their performance, and checking on their credibility of text prediction. The future scope of the best model will be in identifying what a drawn number is and that may help in pointing out students needing more help.

Data Cleaning

The dataset is a record of 42,000 observations under 46 variables, containing the numerical values for pixels of the digits. As the first step, we examined each column of the dataset for missing values. There were no missing values found in the dataset. Additionally, we checked the columns for any special characters, but none were found. We reviewed the data types of all the column variables and found all to be integer values. Among all the given variables, 'Label' will be the target variable for our models.

K Nearest Neighbors (KNN) Model

The KNN model provides a simple approach in predicting the handwritten numbers with the help of k nearest matches in the training data and then using them to predict the target variable’s class. Traditionally, Euclidean distance is used to find the closest match**.** The dataset is split into training and testing sets in 80:20 ratio. The training set will be used to train the KNN model, and the testing set will be used to evaluate its accuracy.

Model 1:

Deciding the value of k in the KNN (K-Nearest Neighbors) model requires consideration of several factors. This ensures that ties between the nearest neighbors can be resolved by majority voting. A common rule of thumb is to set k as the square root of the total number of samples in the dataset. Since there are 33600 samples in this dataset, the square root value is approximately 205.

Model Performance: When all the variables are included in the model to predict the handwriting and ‘k’ is chosen to be equal to 205, the accuracy of the model is roughly 61.12%. ‘Precision’ tells us that out of all the predictions, 82% predictions were accurate for handwritten number ‘6’, whereas ‘Recall’ tells us that out of all the responses under each class, here, the precision and recall values for’0’ and ‘6’ are predicted most accurately with 86% and 81% recall value respectively. The F1 score confirms that this model will perform better at accurately predicting the handwriting of label ‘6’.

Model 2:

In general, as the number of classes increases, it is advisable to use larger values of K to ensure a more robust decision based on a larger pool of neighbors. However, it is important to avoid excessively large values of K, which may introduce noise and increase chances of misclassification. Moreover, higher values of K require more computational resources and time for prediction, as they involve more distance calculations. Considering the trade-off between model performance and computational efficiency, we experimented with a range of values of k on the train data and chose a significantly small value of 103.

Model Performance: When all the variables are included in the model to predict the handwriting and 'k' value is reduced to 103, the accuracy of the model increases to 63.38%. 'Precision' tells us that out of all the predictions, 83% predictions were accurate for handwritten number '6', whereas 'Recall' tells us that out of all the responses under each class, here, the precision and recall values for'1' and '6' are predicted most accurately with 97% and 85% recall value respectively. The F1 score confirms that this model will perform better at accurately predicting the handwriting of label '6' with 0.84 score.

Model comparison: Both models confirm that the KNN model is better trained and more accurate in predicting the label '6'. However, the accuracy of the model with a smaller k value is better at 63.4% compared to the first model with a higher k. As per precision, both models suggest the most accurate prediction of label '6' using KNN. As per Recall value, both models suggest better prediction of label '1'. The F1 score suggests that KNN can predict label '6' better than all the other classes of the target variable. Also, the value of k should be better set at 103 over 205 in terms of model performance.

Challenges:

Handwriting recognition often involves images having high-dimensional data due to large number of features (pixels). High-dimensionality of data increases computational complexity and potentially diminishing the KNN algorithm's performance. The choice of 'k' can significantly impact the accuracy of the model and this choice becomes even more critical while dealing with multiple classes. A small value of k may result in overfitting, while a large value may introduce bias and ignore local patterns by including samples from other classes in the nearest neighbors and causing misclassifications. It is a challenging task to experiment with different values of k is essential to find the balance between underfitting and overfitting. Handwriting can display many variations due to factors like different writing styles, ink smudges, or pen pressure variations. This acts like noise in the KNN model and makes it challenging for the algorithm to capture inherent patterns.

Random Forest Model

Using random forest classification, multiple decision trees are created from different random subsets of the train dataset. As a starting point, we set n_estimators, that is, the number of decision trees in the forest, to 100 and then move on to the next model with 200. Since predictions for the 'label' of handwritten numbers are made by calculating the prediction for each decision tree and taking the most popular result, each decision tree in the Random Forest contributes to the result. Therefore, changing the number of trees in the Random Forest model should affect the accuracy in some way.

Model 1:

In this model, Predictions for the ‘label’ class are made using the hyperparameter, ‘n_estimators’, that is, the number of decision trees in the forest, tuned to 100 as a baseline level ensuring that the computational cost of training and predicting is not increased and neither is the performance of the model compromised.

Model Performance:

The accuracy of the model is 60%, that is, it correctly predicted the handwriting for approximately 60% of the samples in the test dataset. However, as per ‘Precision’ out of all the predictions, 78% of predictions were actually accurate for the ‘6’ label class, followed by 73% for ‘0’ and 71% for ‘5’. The ‘Recall’ values tell us that out of all the responses under ‘1’ label class, 98% of them were correctly predicted for label ‘1’, followed by 85% for label ‘0’. These precision and recall values favor different classes. The F1 score, being the harmonic mean of the two metrics, suggests that the model is better at predicting label ‘6’ with a higher score at 0.79, followed by label ‘0’ at 0.78 score.

Model 2:

Predictions for the ‘label’ class are made by fixing the hyperparameter ‘n_estimators’ at 200. Since it is higher value than 100, it may improve the performance of the model but at the same time increases the computational cost of training and predicting.

Model Performance: The accuracy of the model is 59.95%, that is, it correctly predicted the handwriting for approximately 60% of the samples in the test dataset. Increasing the number of decision trees to be averaged out in the forest has had no impact on the accuracy whatsoever. As per ‘Precision’ out of all the predictions, 77% of predictions were accurate for the label ‘6’ followed by 74% for label ‘0’. The ‘Recall’ tells us that out of all the predictions under label ‘1’, 98% of them were correctly predicted, followed by 81% for label ‘6’ as well as ‘7’. These precision and recall values favor different classes. The F1 score, being the harmonic mean of the two metrics, suggests that the model is better at predicting the label ‘0’ and label ‘6’, both with the same score of 0.88.

Model comparison: The predictive accuracy level for the second random forest has seen almost no change from a 60% accuracy to a 59.9% accuracy. Thus, increasing the number of trees has not increased model performance, nor it has decreased it by overfitting. Looking at the classification report provided for both the models, we can infer that label ‘6’ has the highest precision, recall, and F1-score among all the classes and is the best-predicted class by both models.

Challenges:

Random Forest models can suffer from overfitting if the number of trees is too high and underfitting if the number of trees is too low. Moreover, a Random Forest model with a higher number of trees requires more memory computationally. Finding the optimal number of trees and the depth of model requires experimentation and tuning, and then evaluating the model's performance on multiple evaluation metrics. This can be a time-consuming and challenging task.

Neural Network Model

We created a Neural network model using the Sequential API, which allows for a linear stack of layers. In both the models, we have used 4 layers but with different activation functions. The input data, X, includes all the features after dropping the 'label' column and the target variable, ‘y’, is the ‘label’ class. To train the model, the data is split into training and testing sets comprising 80% and 20% data respectively.

Model 1

This model chooses ‘softmax’ and ‘ReLU (Rectified Linear Unit)’ activation functions. The ‘softmax’ activation function is considered suitable for multi-class classification problems and is commonly used in the output layer when there are more than two classes. The ‘ReLU’ is more commonly used in hidden layers as it accomodates non-linearity in the model and helps it in learning complex patterns. The model starts with 2 layers with ‘ReLU’ activation function for both. The third layer prevents overfitting with a dropout rate of 0.5. The final layer with 10 neurons (corresponding to the 10 classes in the handwriting prediction task) uses a ‘softmax’ activation function. The number of epochs is set to 20, specifying the number of times the model will iterate over the entire train dataset during model training.

Model Performance: The accuracy improves over epochs, showing that the model is gradually learning to make better predictions. The accuracy achieved by the model on the testing set is 66.79%, from an initial accuracy of around 21.76% (in Epoch 1). It is observed that the loss value, an indicator of how well the model is performing in terms of minimizing the difference between predicted and actual labels, is approx. 0.95 on the training set. However, if we look at the Precision and Recall values of the neural network, we find that the values are 0.88 and 0.90 respectively for the label number ‘6’. Based on F1 score it predicts ‘1’ label is better, with an F1-score of 0.90. This indicates that the model achieved a good balance between precision and recall for label class ‘1’. Additionally, the precision and recall values for class ‘1’ are also relatively high, with precision of 0.84 and recall of 0.96. Therefore, based on the given information, class 1 is the best-predicted class by this model.

Model 2

This model uses the ‘Leaky ReLU’ activation function for the hidden layers. ‘Leaky ReLU’ is a variant of ReLU that avoids the ‘dying ReLU’ problem wherein ReLU neurons can become inactive, leading to gradients falling to zero and allowing no learning during the training process. The activation function for the output layer is still chosen to be ‘softmax’.

Model Performance: It is observed that the loss value on the testing set is approx. 0.9363 and the accuracy achieved by the model on the testing set is 66.45%, from an initial accuracy of around 35.21% (in Epoch 1). This indicates that the model is learning and improving its performance over time. The loss value decreases from 9.4289 in the first epoch to 1.0214 in the final epoch. Lower loss values indicate that the model's predictions are becoming closer to the actual values. The precision is highest for label ‘6’, implying that out of all the combined predictions for numbers 0-9, 85% were accurate for this label. The recall for label ‘6’ is 90% but is highest for label ‘1’ at 96%, indicating that 96% of all prediction for label ‘1’ turned out true. The class with the highest F1-score of 0.87, is label ‘6’. This suggests that the model achieved a good balance between precision and recall for label ‘6’. Based on the F1-score, the best-predicted class would be class 6.

<u>Model comparison:</u> The predictive accuracy level for the first neural network is slightly higher than the second model. Thus, changing the activation function for the hidden layers from ‘ReLU’ to ‘Leaky ReLU’ has not made a considerable improvement and in fact has reduced the accuracy by 0.34. Looking at the classification report provided for both the models, we can infer that label ‘6’ has the highest precision, recall, and F1-score among all the classes and is the best-predicted class by both models.

Challenges:

Neural networks are prone to overfitting, especially when dealing with limited training data, therefore a dropout layer was added with a dropout rate of 0.5. Additionally, training neural networks, requires significant computational time. Hyperparameter Tuning: Selecting the optimal hyperparameters, including activation functions, learning rate, batch size, and number of layers, can be challenging. It often requires experimentation and fine-tuning to find the best combination for maximizing accuracy. Handwriting data may have variations in writing styles, ink smudges, or pen pressure which can affect the model's ability to generalize results to different variations.

Model Comparison (all models)

The table below gives a comprehensive summary of key performance metrics of all the three types of models used in this project. Since the Recall and Precision values, for all the models, favor different label classes, therefore we use F1 score to see which class of ‘Label’ is predicted best by the model.

Model Type	Best Accuracy Achieved	Highest Precision Value Observed	Highest Recall Value Observed	Best F1 Score Achieved	Best ‘Label’ class predicted based on F1 score
KNN (k=103)	63.4%	0.82	0.97	0.84	‘6’
Random Forest (n_estimators= 100)	60%	0.77	0.98	0.79	‘6’
Neural Network (activation functions: ‘ReLU’ and ‘softmax’)	66.79%	0.88	0.96	0.90	‘1’

Conclusion

The handwriting test analysis in this project includes three different types of models. It is recommended that the Neural Network model is the most appropriate one for predicting students' handwriting of numbers 0-9. Subsequently, this model can become a stepping-stone to identify students who could need motor skill development. The Neural Network consistently outperformed the KNN and Random Forest models in accuracy, precision, and the F1 score. All the three models show that this dataset can be used best to predict the handwritten labels ‘6’ and ‘1’ better than rest of all the 10 label classes. According to the Neural network, this dataset gives significantly better results in predicting number label ‘1’ over ‘6’. Overall, based on this dataset, we can get reliable predictions for number labels ‘0’, ‘1’ and ‘6’ using any of the models in this project. However, the pixel intensity data used here falls a bit short in predicting numbers 2,3,4,5,7,8, and 9. It performs most poorly in predicting '9' and this needs to be improved.

Sharing this neural network model with teachers and parents allows for targeted interventions to enhance students' motor skills from an early age, since early identification of motor skill difficulties improves academic performance and boosts confidence. This data-driven approach is expected to give reliable results since this involves evidence-based decision-making by the school management.

References

Python, Real. "Logistic Regression in Python – Real Python." *Realpython.com*, realpython.com/logistic-regression-python/

"Guide to Accuracy, Precision, and Recall." *Mage*, www.mage.ai/blog/definitive-guide-to-accuracy-precision-recall-for-product-developers.

Shafi, Adam. "Sklearn Random Forest Classifiers in Python Tutorial." *Www.datacamp.com*, Feb. 2023, www.datacamp.com/tutorial/random-forests-classifier-python.

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

CS50. (2020, April 10). *Neural Networks - Lecture 5 - CS50's Introduction to Artificial Intelligence with Python 2020* [Video]. YouTube. https://www.youtube.com/watch?v=mFZazxxCKbw

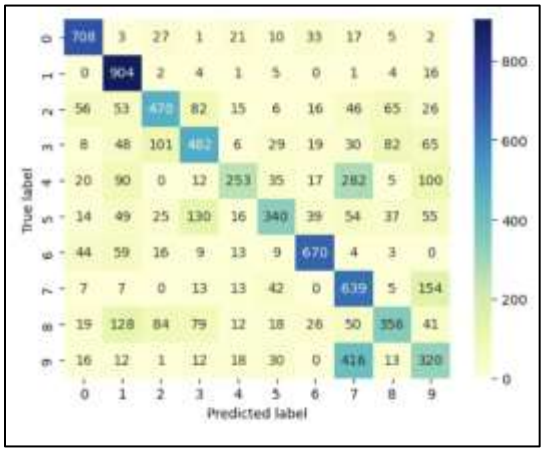
Aggarwal, S. (2021, December 14). K-Nearest Neighbors - Towards Data Science. *Medium*. https://towardsdatascience.com/k-nearest-neighbors-94395f445221

<u>Appendix</u>

KNN Model

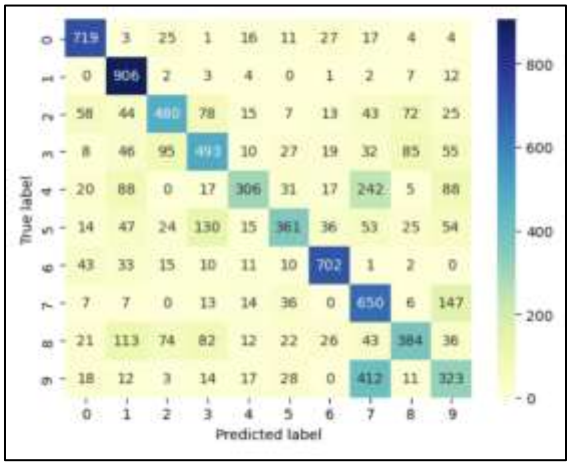
<u>Confusion Matrix:</u>

Model 1:



	0	1	2	3	4	5	6	7	8	9
0	708	3	27	1	21	10	33	17	5	2
1	0	904	2	4	1	5	0	1	4	16
2	56	53	470	82	15	6	16	46	65	26
3	8	48	101	482	6	29	19	30	82	65
4	20	90	0	12	253	35	17	282	5	100
5	14	49	25	130	16	340	39	54	37	55
6	44	59	16	9	13	9	670	4	3	0
7	7	7	0	13	13	42	0	639	5	154
8	19	128	84	79	12	18	26	50	356	41
9	16	12	1	12	18	30	0	416	13	320

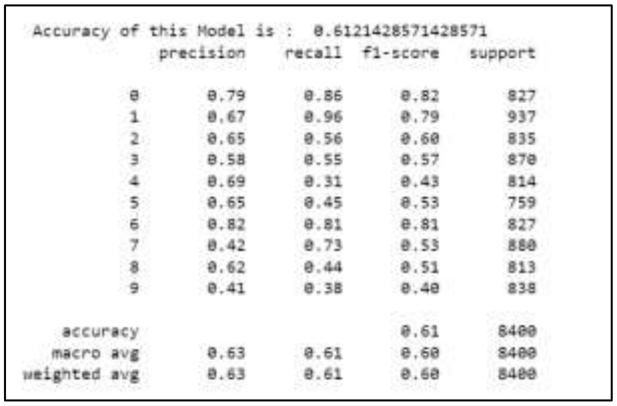
Model 2:



	0	1	2	3	4	5	6	7	8	9
0	719	3	25	1	16	11	27	17	4	4
1	0	906	2	3	4	0	1	2	7	12
2	58	44	480	78	15	7	13	43	72	25
3	8	46	95	493	10	27	19	32	85	55
4	20	88	0	17	306	31	17	242	5	88
5	14	47	24	130	15	361	36	53	25	54
6	43	33	15	10	11	10	702	1	2	0
7	7	7	0	13	14	36	0	650	6	147
8	21	113	74	82	12	22	26	43	384	36
9	18	12	3	14	17	28	0	412	11	323

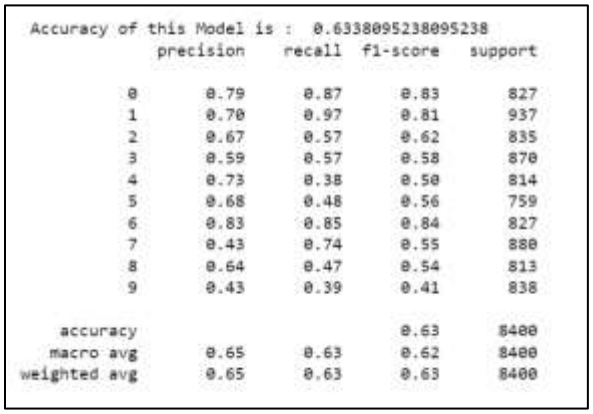
<u>Performance Metrics:</u>

Model 1:



```
Accuracy of this Model is :  0.6121428571428571
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       827
           1       0.67      0.96      0.79       937
           2       0.65      0.56      0.60       835
           3       0.58      0.55      0.57       870
           4       0.69      0.31      0.43       814
           5       0.65      0.45      0.53       759
           6       0.82      0.81      0.81       827
           7       0.42      0.73      0.53       880
           8       0.62      0.44      0.51       813
           9       0.41      0.38      0.40       838

    accuracy                           0.61      8400
   macro avg       0.63      0.61      0.60      8400
weighted avg       0.63      0.61      0.60      8400
```

Model 2:



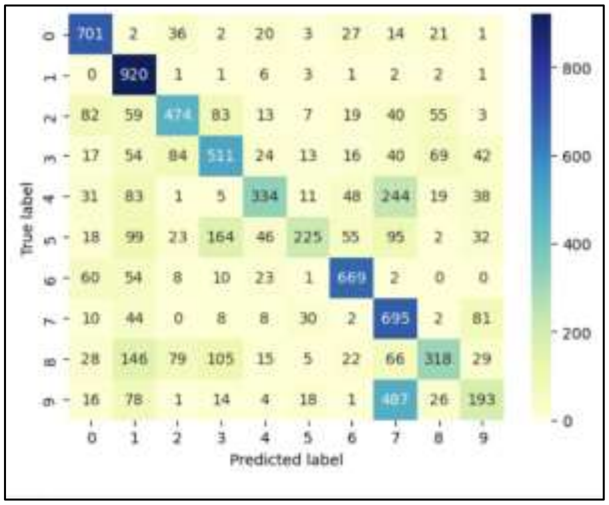
```
Accuracy of this Model is :  0.6338095238095238
              precision    recall  f1-score   support

           0       0.79      0.87      0.83       827
           1       0.70      0.97      0.81       937
           2       0.67      0.57      0.62       835
           3       0.59      0.57      0.58       870
           4       0.73      0.38      0.50       814
           5       0.68      0.48      0.56       759
           6       0.83      0.85      0.84       827
           7       0.43      0.74      0.55       880
           8       0.64      0.47      0.54       813
           9       0.43      0.39      0.41       838

    accuracy                           0.63      8400
   macro avg       0.65      0.63      0.62      8400
weighted avg       0.65      0.63      0.63      8400
```

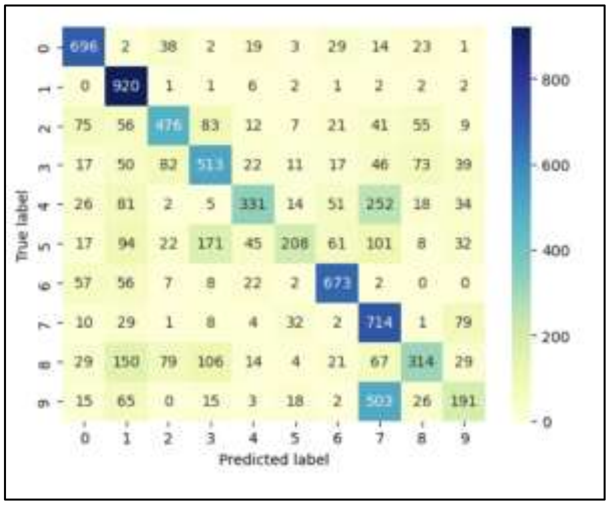
Random Forest Model

<u>Confusion matrix:</u>

Model 1:

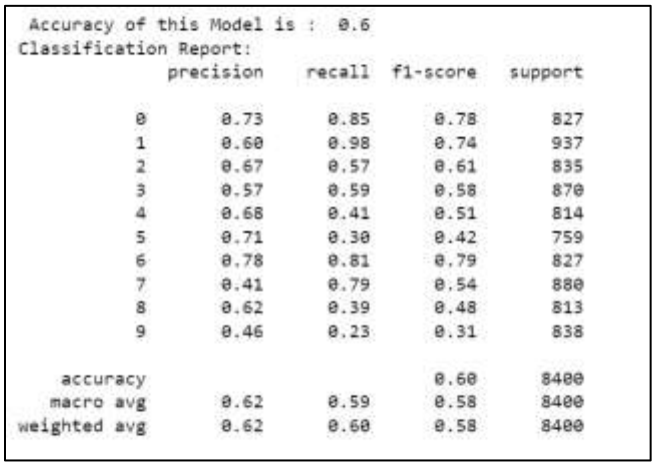
Model 2:





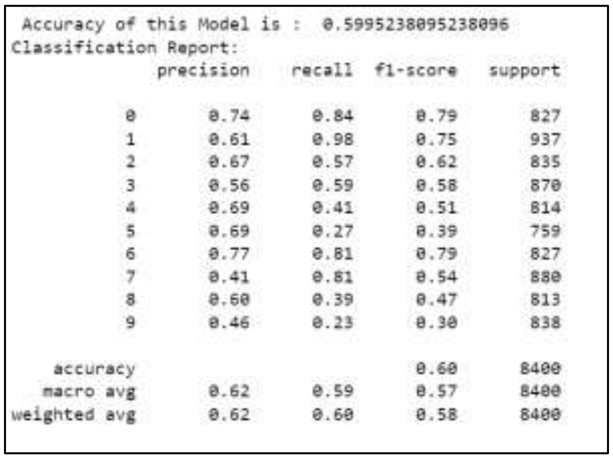
Performance Metrics:

Model 1:



```
 Accuracy of this Model is :  0.6
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.85      0.78       827
           1       0.60      0.98      0.74       937
           2       0.67      0.57      0.61       835
           3       0.57      0.59      0.58       870
           4       0.68      0.41      0.51       814
           5       0.71      0.30      0.42       759
           6       0.78      0.81      0.79       827
           7       0.41      0.79      0.54       880
           8       0.62      0.39      0.48       813
           9       0.46      0.23      0.31       838

    accuracy                           0.60      8400
   macro avg       0.62      0.59      0.58      8400
weighted avg       0.62      0.60      0.58      8400
```

Model 2:



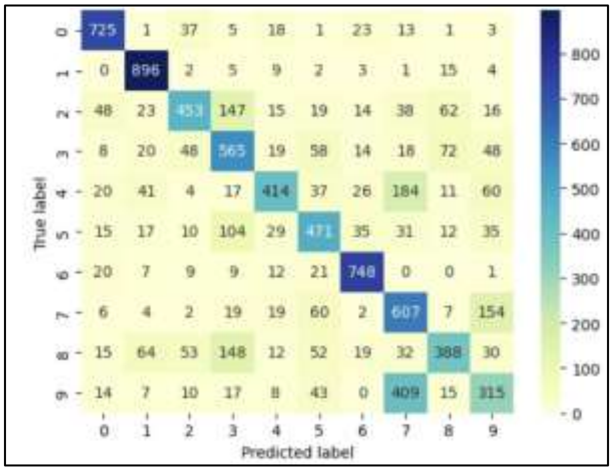
```
 Accuracy of this Model is :  0.5995238095238096
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.84      0.79       827
           1       0.61      0.98      0.75       937
           2       0.67      0.57      0.62       835
           3       0.56      0.59      0.58       870
           4       0.69      0.41      0.51       814
           5       0.69      0.27      0.39       759
           6       0.77      0.81      0.79       827
           7       0.41      0.81      0.54       880
           8       0.60      0.39      0.47       813
           9       0.46      0.23      0.30       838

    accuracy                           0.60      8400
   macro avg       0.62      0.59      0.57      8400
weighted avg       0.62      0.60      0.58      8400
```

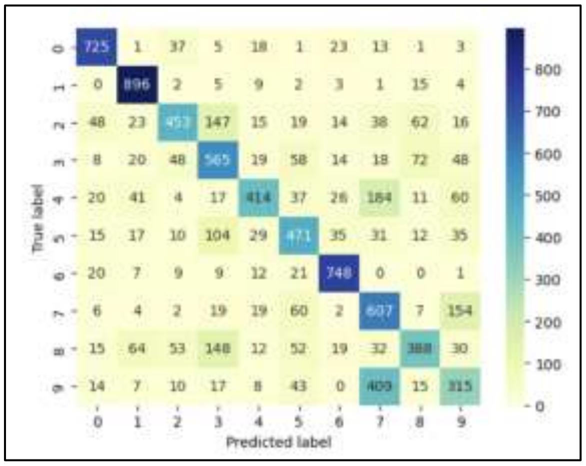
Neural Network

Confusion Matrix:

Model 1:

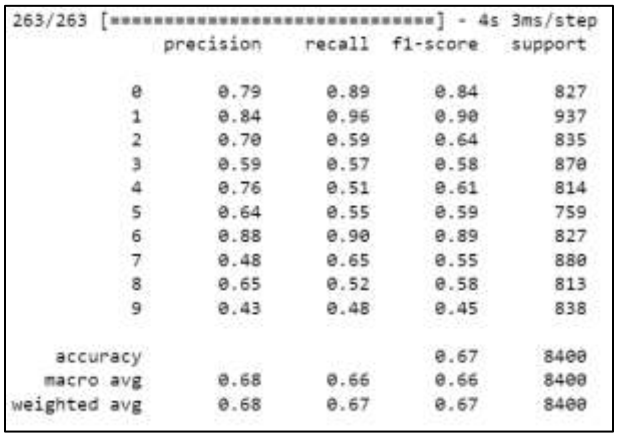
Model 2:





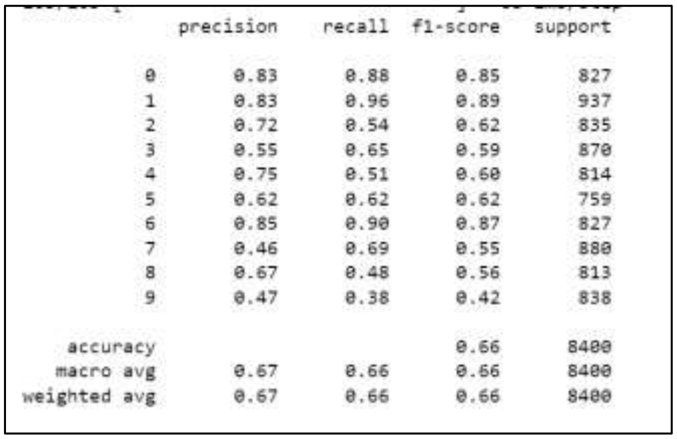
<u>Performance Metrics:</u>

Model 1:



```
263/263 [==============================] - 4s 3ms/step
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       827
           1       0.84      0.96      0.90       937
           2       0.70      0.59      0.64       835
           3       0.59      0.57      0.58       870
           4       0.76      0.51      0.61       814
           5       0.64      0.55      0.59       759
           6       0.88      0.90      0.89       827
           7       0.48      0.65      0.55       880
           8       0.65      0.52      0.58       813
           9       0.43      0.48      0.45       838

    accuracy                           0.67      8400
   macro avg       0.68      0.66      0.66      8400
weighted avg       0.68      0.67      0.67      8400
```

Model 2:



```
              precision    recall  f1-score   support

           0       0.83      0.88      0.85       827
           1       0.83      0.96      0.89       937
           2       0.72      0.54      0.62       835
           3       0.55      0.65      0.59       870
           4       0.75      0.51      0.60       814
           5       0.62      0.62      0.62       759
           6       0.85      0.90      0.87       827
           7       0.46      0.69      0.55       880
           8       0.67      0.48      0.56       813
           9       0.47      0.38      0.42       838

    accuracy                           0.66      8400
   macro avg       0.67      0.66      0.66      8400
weighted avg       0.67      0.66      0.66      8400
```