Northeastern University

Week 5 Assignment

Course: ALY 6020 PREDICTIVE ANALYSIS
Academic Year: Winter 2023

Instructor: JUSTIN

Submitted by: Manas Babbar
Submission Date: March 26th, 2023

INTRODUCTION

Business Problem: The school is trying to identify students who may need help with their motor skills at a young age. They want to use a writing test to determine if students are struggling with writing numbers. They believe that if they can accurately predict what students have written, they can **identify those who need help** with their motor skills. The school is interested in building a model that can accurately classify what students have written.

The ability to write numbers is an important skill for students to develop, and it can be an indicator of their motor skills development. The school is interested in using a writing test to identify students who may need help with their motor skills at a young age. In this project, we will be building a **classification model using KNN, Random forest and neural networks** to predict what students have written based on the numbers they have drawn. The dataset provided contains images of numbers that students have written, along with labels indicating what number is being represented. We will be using this dataset to train and evaluate our models. Our goal is to build a model that can accurately classify what students have written, with the hope of identifying those who may need extra help with their motor skills.

We will be comparing the performance of KNN, Random forest and neural networks to see which model performs better on this task. The school is interested in using a simplistic approach like KNN to start the research, but we will also be exploring more complex models like neural networks. Ultimately, the goal is to find a model that can accurately predict what students have written, and use this information to identify those who may need extra help.

Dataset description: The dataset contains images of handwritten digits (0 to 9) with their respective labels. It has **42000 rows** and **46 columns**. The first column ('label') contains the labels, while the remaining columns ('pixel43', 'pixel44', etc.) contain the pixel values of each image. Each image is a 28x28 pixel grayscale image, and each pixel is represented by an integer value between 0 and 255, inclusive.

ANALYSIS

Null & Duplicate Values: The first thing I did is check for duplicate and null values in our dataset and found out that there were no null values present in the dataset but there were **1633** Duplicate rows in the dataset which were removed as it could harm our analysis and result in a bias for our predictive models.

Created a **Bar plot** to check the distribution of labels which is our target variable for the classification models. Through the bar plot, we could see that this is a case of balanced classification in the dataset as there were no high relative differences in the count of each class. Checking for **classification imbalance** is important before making the model because an imbalanced dataset can result in poor performance of the classification model. When a dataset is imbalanced, the model tends to be biased towards the majority class and may neglect the minority class, leading to poor predictive accuracy on the minority class.

Normalising the data is also a very important step because it ensures that each feature contributes equally to the distance calculations and prevents features with larger scales from dominating the model. This is especially important for distance-based algorithms such as k-nearest neighbours, clustering algorithms, and support vector machines. In this problem, the **standard scaler** was used to normalize the data because it transforms the features to have zero mean and unit variance. This means that the data will have a normal distribution with a mean of zero and a standard deviation of one, which makes it easier for the machine learning model to learn the underlying patterns in the data. Additionally, the standard scaler is less affected by outliers compared to other normalization techniques.

K-Nearest Neighbours

It is a simple classification algorithm that classifies new data points based on the majority class of its k nearest neighbours in the training set. The value of k is determined by the user and it impacts the model's performance. KNN is a non-parametric and instance-based algorithm that does not require assumptions about the distribution of the data. The accuracy that we got in this Model was **0.656** means that the model correctly predicted 65.6% of the classes in the training set. We made a confusion matrix for the cross-validation of our model.

The precision, recall, and f1-score for each class vary, with some classes performing better than others. For instance, the model performed **well in predicting classes 0, 1, and 6,** with f1-scores of 0.85 or higher. On the other hand, the model had **lower performance in predicting classes 3, 4, 7, 8, and 9,** with f1-scores ranging from 0.46 to 0.58.

Challenge was that KNN requires the user to specify the value of K, the number of nearest neighbours to consider. Choosing the right value of K can be challenging, and it can have a significant impact on the performance of the model. A small value of K can lead to overfitting, while a large value of K can lead to underfitting. So to get the best KNN value I used the trial method with possible values and used the one which was giving the highest accuracy.

The MSE value for this model is 5.0.

Random Forest

Machine learning algorithm used for classification and regression tasks. It operates by creating multiple decision trees and averaging the results to improve accuracy and reduce overfitting. Each tree is built using a random subset of the features and a subset of the training data, making the model more robust and less sensitive to outliers. The accuracy that we got in this Model was **0.708** means that the model correctly predicted 70.8% of the classes in the training set. We made a confusion matrix for the cross-validation of our model. we can see that the **model performed well in predicting classes 0, 1, and 6,** with f1-scores of 0.88 or higher. This means that the model correctly predicted the majority of instances belonging to these classes, and that the precision and recall for these classes were both high. On the other hand, the model had **lower performance in predicting classes 3, 4, 7, 8, and 9**, with f1-scores ranging from 0.48 to 0.66. This means that the model struggled to correctly predict instances belonging to these classes, and that the precision and recall for these classes were lower. Overall, while the random forest model performed better than the KNN model in terms of overall accuracy, it still had some difficulties in correctly predicting instances for certain classes. The **Feature of most importance** for this model was the **pixel 329.** This could be because that pixel contains a distinctive pattern or feature that is unique to a certain class, and the model relies heavily on that information to make accurate predictions.

One challenge that I faced for the random forest classification model is that it required more resources and time to train compared to simpler models such as KNN or decision trees. Additionally, random forests looked to be prone to overfitting if the model is too complex or if the number of trees in the forest is too high. It can also be difficult to interpret the model and understand the relative importance of each feature in making predictions.

The MSE value for this model is 4.13

Neural Network

It is a class of machine learning models inspired by the structure and function of biological neural networks in the human brain. They consist of interconnected layers of nodes that process and transform data, allowing them to learn and make predictions from complex patterns in large datasets. The accuracy that we got in this Model was **0.708** means that the model correctly predicted 70.8% of the classes in the training set. We made a confusion matrix for the cross-validation of our model. The model **performed well in predicting classes 0, 2, and 3,** with f1-scores of 0.88 or higher. This means that the model correctly predicted the majority of instances belonging to these classes and that the precision and recall for these classes were both high. On the other hand, the model had **lower performance in predicting classes 1 and 4,** with f1-scores ranging from 0.59 to 0.74. This means that the model struggled to correctly predict instances belonging to these classes and that the precision and recall for these classes were lower. Overall, while the Random forest model had a relatively high overall accuracy, it still had some difficulties in correctly predicting instances for certain classes.

The **challenge** I faced in this Model is the selection of appropriate hyperparameters such as the learning rate, number of layers, and number of neurons in each layer. Additionally, the choice of the appropriate loss function and activation function also impacted the performance of the model.

The MSE value for this model is 4.51

COMPARISON OF THESE MODEL

Model	Accuracy	Precision (macro avg)	Recall (macro avg)	F1-Score (macro avg)	MSE
Neural Network	0.69	0.70	0.68	0.68	4.51
KNN	0.66	0.65	0.65	0.64	5.00
Random Forest	0.71	0.71	0.70	0.70	4.13

1. Accuracy: The accuracy of the Random Forest model is 71%, while the accuracy of the KNN model is 66% and NN is 69%. This suggests that the Random Forest model is able to make more correct predictions than the KNN and NN models.
2. Precision and Recall: In general, the precision and recall values for the Random Forest model are higher than those of the KNN model. For example, the precision for class 0 in the Random Forest model is 85%, while the precision for class 0 in the KNN model is 82%. Similarly, the recall for class 4 in the Random Forest model is 57%, while the recall for class 4 in the KNN model is 50%. Higher precision and recall values indicate that the model is better able to correctly identify the samples belonging to each class.
3. F1-score: The F1-score is a metric that takes into account both precision and recall. In general, the F1-scores for the Random Forest model are higher than those of the KNN model. For example, the F1-score for class 0 in the Random Forest model is 88%, while the F1-score for class 0 in the KNN model is 85%. Higher F1-scores indicate that the model is better able to balance precision and recall.

CONCLUSION

Based on the evaluation metrics and performance of the different models, it can be concluded that the **Random Forest performed the best** among the models we trained. Models achieved **high accuracy** and low errors, with the Random Forest being the **best also in terms of Mean squared error (MSE)**. It is important to note that the Random Forest Regressor had a shorter training time compared to others. Therefore, if speed is a critical factor

Also, we could see some Models were good in predicting some labels as compared to others so we can also use a combination of the model by selecting the labels they performed best for. Furthermore, if interpretability is important, then the KNN model can be considered. Although it did not perform as well as the other models in terms of accuracy,. But in case we could gather a lot more data maybe this will not have this advantage and neural network will be preferred. it has the advantage of being easy to interpret and understand. In general, the choice of model to use will depend on the specific problem and context. If high accuracy is the top priority and time then Random Forest Regressor can be used.

It is recommended to use a variety of models and compare their performance before deciding on the final model to use. In addition, it may be beneficial to combine the predictions of multiple models using techniques such as ensembling or stacking to improve the overall performance.

APPENDIX

Support file As@ipynb

Fig 1

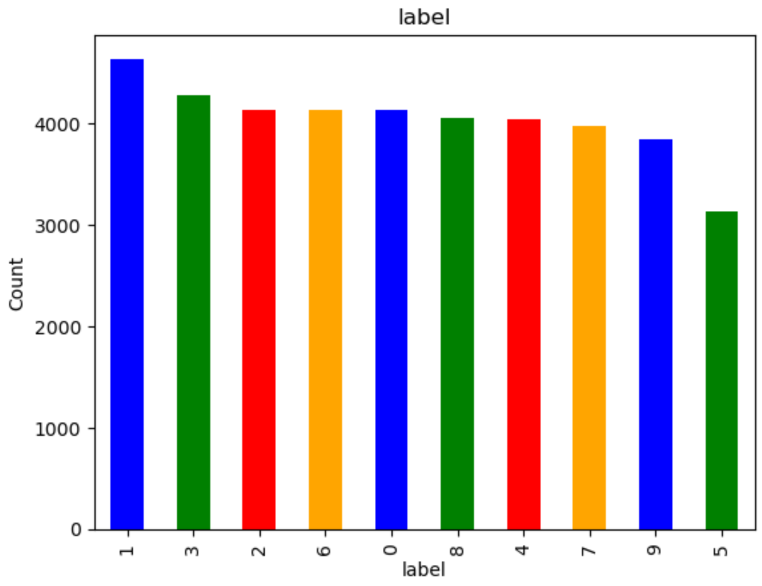

Fig 2

```
[[766   1  27   4  14   3  23  15   2   9]
 [  0 855   8   5   6   7   7   3  10   1]
 [ 57  18 532  79  17   8  25  32  53  19]
 [ 13  29 119 496  17  35   6  23  85  33]
 [ 22  74   8  14 413  25  18 154  13  82]
 [ 10  22  20 115  27 290  39  36  15  47]
 [ 23  18  21  10  10  20 699   0   3   0]
 [ 10   5   7  13  48  21   1 497  10 160]
 [ 13  68  86 105  14  38  17  32 408  30]
 [ 21  11   8  24  30  26   0 288  30 343]]
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       864
           1       0.78      0.95      0.85       902
           2       0.64      0.63      0.63       840
           3       0.57      0.58      0.58       856
           4       0.69      0.50      0.58       823
           5       0.61      0.47      0.53       621
           6       0.84      0.87      0.85       804
           7       0.46      0.64      0.54       772
           8       0.65      0.50      0.57       811
           9       0.47      0.44      0.46       781

    accuracy                           0.66      8074
   macro avg       0.65      0.65      0.64      8074
weighted avg       0.66      0.66      0.65      8074
```

Fig 3

```
[[785   0  26   3  12   1  14  14   5   4]
 [  0 861   5   4   5   3   5   2  12   5]
 [ 53  14 560  59  25   6  20  26  63  14]
 [ 13  17  62 548  16  40   8  19  92  41]
 [ 14  43   5   7 470  12  29 140  13  90]
 [  9  12   6  89  24 342  44  40  12  43]
 [ 13   3   7   5  11  13 750   0   2   0]
 [  7   3   5  13  22   4   1 543   5 169]
 [ 10  28  64 101  10  24  16  33 480  45]
 [ 17   9   8  20  17  10   0 296  25 379]]
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       864
           1       0.87      0.95      0.91       902
           2       0.75      0.67      0.71       840
           3       0.65      0.64      0.64       856
           4       0.77      0.57      0.66       823
           5       0.75      0.55      0.64       621
           6       0.85      0.93      0.89       804
           7       0.49      0.70      0.58       772
           8       0.68      0.59      0.63       811
           9       0.48      0.49      0.48       781

    accuracy                           0.71      8074
   macro avg       0.71      0.70      0.70      8074
weighted avg       0.72      0.71      0.71      8074
```

Fig 4

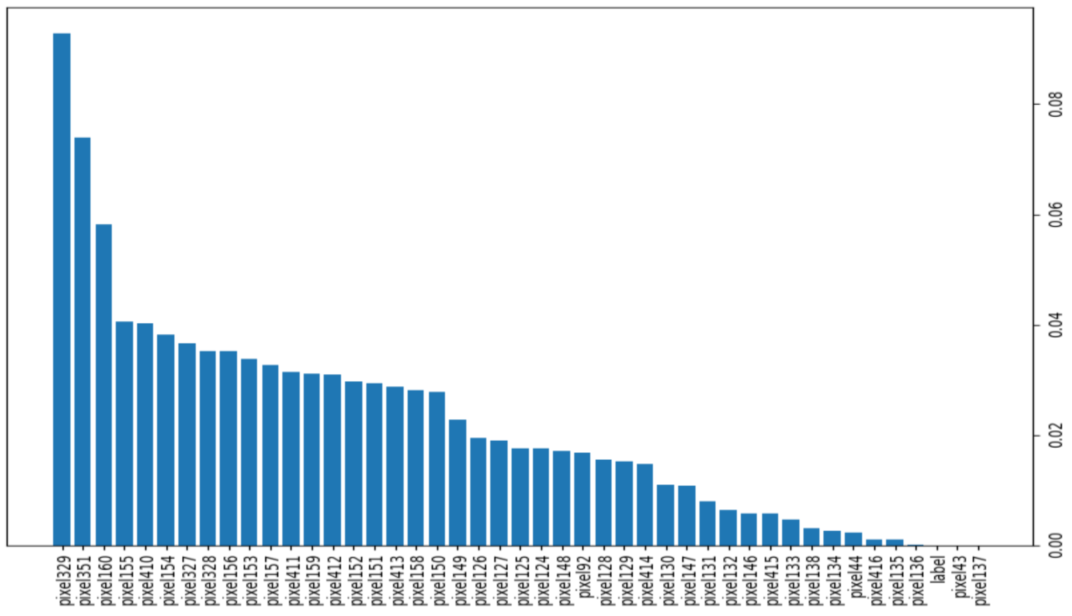

Fig 5

```
[[734   2  51   6   8  11  18  12   7  15]
 [  1 852   4   3   5   8   5   2  16   6]
 [ 33  11 529 100  12  14  14  29  81  17]
 [  8  14  63 512  10  61   8  24 116  40]
 [ 16  45  10   9 436  19  25 149  27  87]
 [  7   8   8  78  15 350  46  47  25  37]
 [ 10   8   7   4  19  27 724   0   5   0]
 [  3   4   8  14  25   6   0 540  10 162]
 [  6  32  57 115   3  31  10  21 497  39]
 [ 18   8   8  20  11   8   0 279  31 398]]
              precision    recall  f1-score   support

           0       0.88      0.85      0.86       864
           1       0.87      0.94      0.90       902
           2       0.71      0.63      0.67       840
           3       0.59      0.60      0.60       856
           4       0.80      0.53      0.64       823
           5       0.65      0.56      0.61       621
           6       0.85      0.90      0.88       804
           7       0.49      0.70      0.58       772
           8       0.61      0.61      0.61       811
           9       0.50      0.51      0.50       781

    accuracy                           0.69      8074
   macro avg       0.70      0.68      0.68      8074
weighted avg       0.70      0.69      0.69      8074
```

Refrences:

1. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer-Verlag New York.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
3. James, G., Witten, D., Hastie, T., & Tibshirani, in python. (2013). An Introduction to Statistical Learning: with Applications in R. Springer.