Introduction

The goal of the assignment is to use machine learning models to predict if a student needs to work on their motor skills. A school is trying to decide if they can use a writing test to understand which students may need to work on their motor skills at a young age. They are putting this to the test by having the students write numbers, and they want to develop a model to see if we can predict what they have drawn. To do this, I'll start my project with a straightforward method (KNN), but then compare the outcomes with something more intricate, like random forest and neural networks. I'll be using various python packages such as numpy, pandas, matplotlib, bamboolib, sklearn, seaborn, scipy, etc.

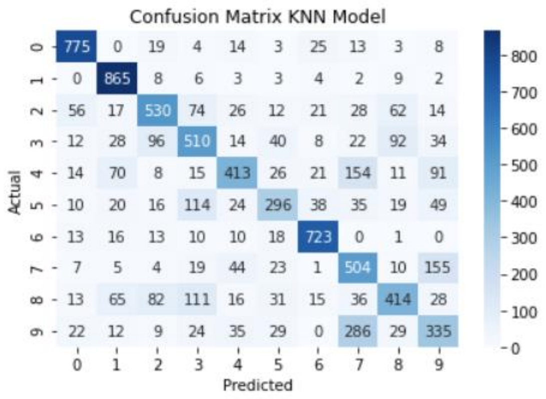
Data Cleaning & EDA

The data set contains 42000 rows and 46 columns. The target columns for the analysis in this project is label and rest 45 columns starting with pixel43 to pixel417 are independent variables. The first step is to check for duplicates and missing values in the data set. There were 1633 duplicates in this data so I dropped them keeping the first ones. There are no missing values in the dataset. Moving on to next step i.e., removing any special characters from the dataset. The datatype of all the variable is integer so no need to update that. As the data is already in numeric format, we do not need to do the label encoding for this type of data to run a classification model. The target variable has values ranging from 0 to 9. So, this is classification problem, that we're trying to solve using a KNN, random forest and neural network models. The histogram of target variable values is displayed in appendix. Next, I plotted a correlation matrix to check for multicollinearity and detect how variables are linked to each other.

KNN Model

KNN (k-Nearest Neighbors) is a machine learning algorithm that can be used for classification or regression tasks. In the context of motor skill testing in schools, KNN can be used as a classifier to predict the skill level of a student based on their performance on a motor skill test. KNN works by finding the k closest data points in the training set to the input data point, and then predicting the label or value of the input based on the labels or values of those k nearest neighbors. In the case of motor skill testing, the input data point would be the performance of a student on the test, and the label would be their skill level.

Now first step to run the KNN model is to split the data in to test and train for further processing. I've divided the data into 80:20 ratio using the train_test_split function from sklearn library. Now taking the n neighbors as 10 in the KNeighborsClassifier, I've created the K nearest predictive model and fitted the values against it. The accuracy of the model is 66.4%. I started off with n neighbors as 2 and gradually increased the value to see if there was any improvement in the accuracy of the model. For n=2 the accuracy was only 60% and as I increased the n value the accuracy also improved for the model. Finally keeping n=10 and with 66.4% model accuracy, I plotted the confusion matrix to check the performance of the model. The below confusion matrix plot shows the actual vs predicted values of the model.

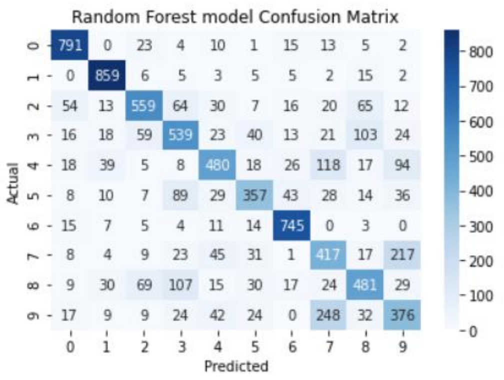


As we can see the confusion matrix is little different from the ones that we're usually used to in most of the machine learning models. Since, this is not just a yes or no prediction problem, the output ranges from 0 to 9 as a result the confusion matrix is also a little different. Each column in the matrix represents the instances in a predicted class, while each row represents the instances in an actual or true class. In this matrix, the diagonal elements represent the number of correctly classified instances, while the off-diagonal elements represent the misclassified instances.

If we look at the top left corner, we can see that there are 775 values that the model predicted to be '0' and are actually '0'. There are 865 values that the model predicted to be 1 and are actually 1. Similarly, there are zero values that the model predicted to be 1 but actually they were 0. The least number of correct predictions were for the number 5 as 296 count while the most correct predictions were for number 2 at 865 count. The most incorrect predictions were made when the model predicted the value to be number 7 but in reality, it was number 9. The classification report in the appendix shows the precision, recall f1-score for each value from 0 to 9 for the KNN model along with the overall model statistics.

Random Forest Model

Next, I created a random forest model using the RandomForestClassifier from sklearn library to check if I can improve the overall accuracy of the model. Using the same test and train split in 80:20 ratio and n estimator=100 i.e., the number of decision tree to be used in the model, I created the random forest model. the accuracy of the random forest model is 69.4%, which is slightly better than the KNN model. The confusion matrix below shows the performance of the random forest model.



Each column in the matrix represents the instances in a predicted class, while each row represents the instances in an actual or true class. In this matrix, the diagonal elements represent the number of correctly classified instances, while the off-diagonal elements represent the misclassified instances.

If we look at the top left corner, we can see that there are 791 values that the model predicted to be '0' and are actually '0'. There are 859 values that the model predicted to be 1 and are actually 1. Similarly, there are zero values that the model predicted to be 1 but actually they were 0. The least number of correct predictions were again for the number 5 as 357 counts while the most correct predictions were for number 2 at 859 count. The most incorrect predictions 248 times were made when the model predicted the value to be number 7 but in reality, it was number 9.

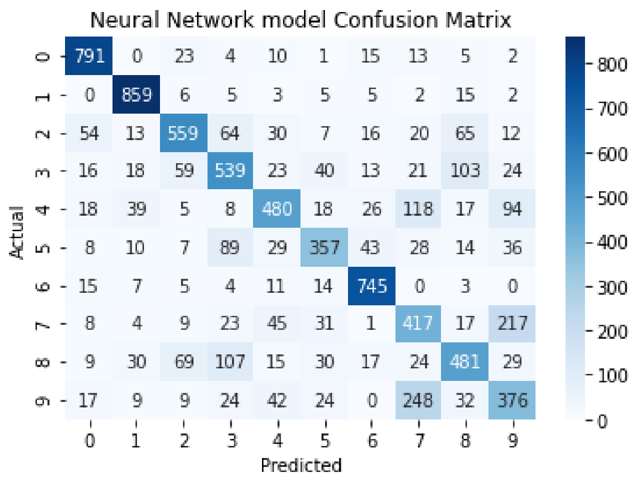
The classification report in the appendix shows the precision, recall f1-score for each value from 0 to 9 for the random forest model along with the overall model statistics. Feature importance graph is displayed in the appendix. Pixel351 has the highest importance at almost 0.11 out of all the features followed by pixel410 at 0.09 and pixel327 at 0.07. Rest all the features have importance of less than 0.05 each.

The two major challenges with the random forest models are the computational complexity and the overfitting. If they are not correctly calibrated, random forest models may overfit the training set of data. Overfitting, which can lead to bad performance when predicting new, unknown data, happens when the model grows too complicated and is too tightly fitted to the training data. Training random forest models may be computationally costly, especially when utilizing multiple trees or working with huge datasets.

Neural Network Model

Motor skill testing using neural network models can provide objective measurements of a student's performance, which can be useful for identifying areas where they may need additional support or intervention. For this project I've used the MLPClassifier from sklearn python package to run the neural network model. Before starting with the model, I normalized the data using the standard scaler from sklearn library so that we don't any outliers in the neural network model.

Taking the max iteration values as 1000 and solver as 'sgd' I build the neural network model with 69.4% model accuracy. 'sgd' in the solver stands for stochastic gradient descent, which updates the weights of neural network equation using a fixed learning rate and minibatches of the data.



The confusion matrix below shows the performance of the random forest model. Each column in the matrix represents the instances in a predicted class, while each row represents the instances in an actual or true class. In this matrix, the diagonal elements represent the number of correctly classified instances, while the off-diagonal elements represent the misclassified instances.

If we look at the top left corner, we can see that there are 791 values that the model predicted to be '0' and are actually '0'. There are 859 values that the model predicted to be 1 and are actually 1. Similarly, there are zero values that the model predicted to be 1 but actually they were 0. The least number of correct predictions were again for the number 5 as 357 counts while the most correct predictions were for number 2 at 859 counts.

The most incorrect predictions 248 times were made when the model predicted the value to be number 7 but in reality, it was number 9. The classification report in the appendix shows the precision, recall f1-score for each value from 0 to 9 for the KNN model along with the overall model statistics. As per the neural network confusion matrix the predicted values are exactly same as that of the random forest model.

Model Comparison

The table below displays the benchmarking statistics of the KNN, random forest and the neural network model.

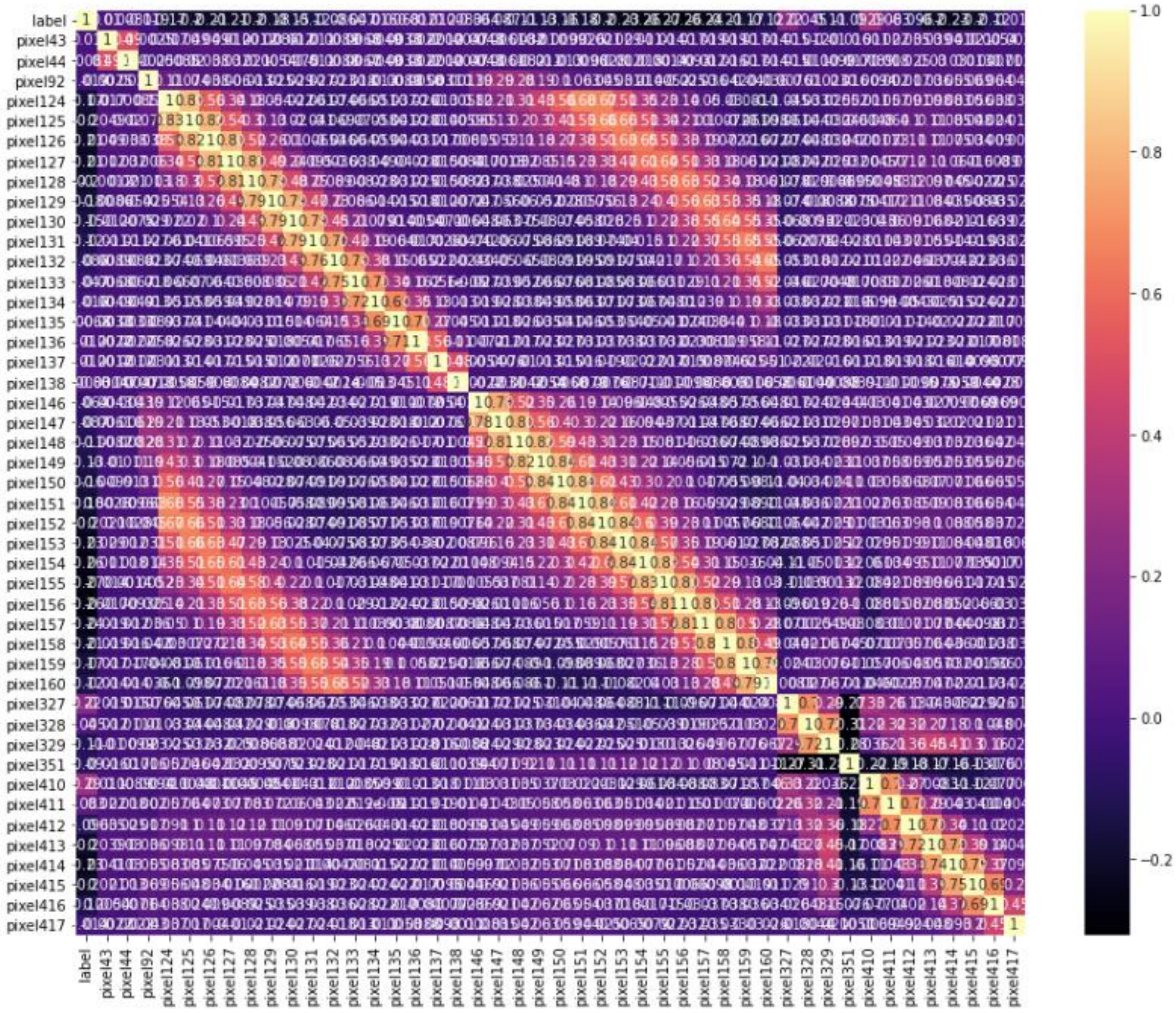
	KNN	**Random Forest**	**Neural network**
Accuracy	66.4%	69.4%	69.58%
Precision	66%	69.3%	69%
Recall	66%	69.4%	69%

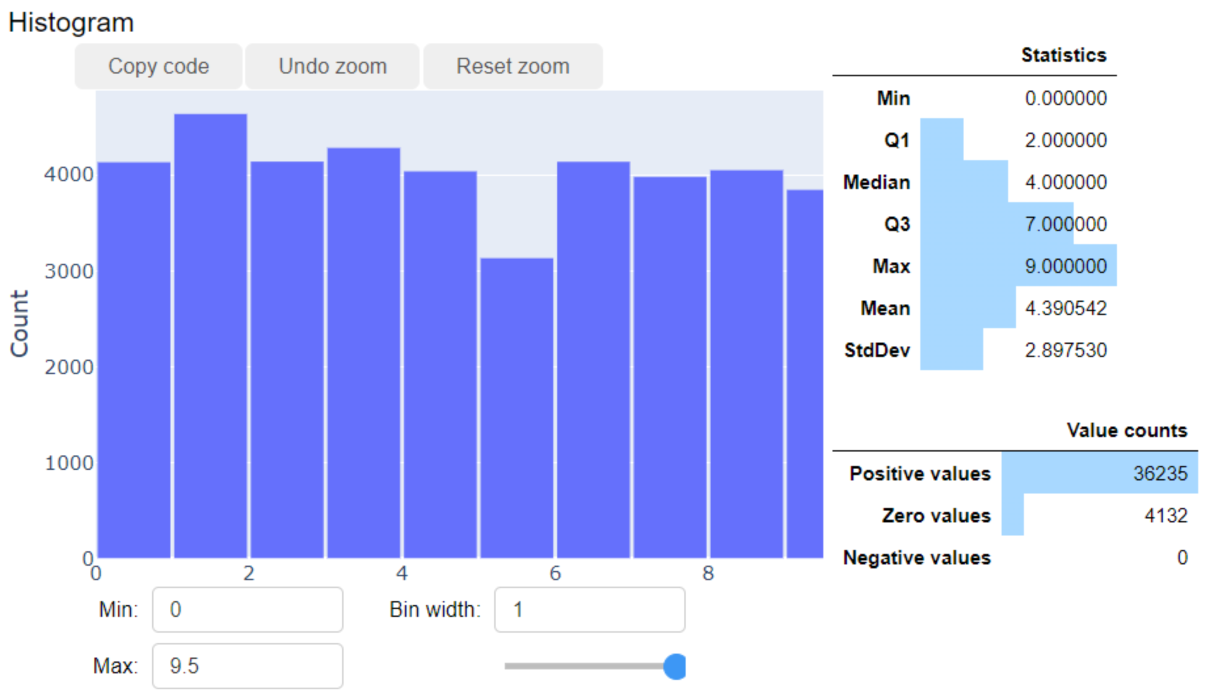
The accuracy is highest for the neural network model at 69.58% closely followed by random forest model and the KNN model has the least accuracy score at 66.4%.

As per the above benchmarking statistics table the best accuracy for the model is for neural networks. Since this model is to be used to test the motor skills for students, I would recommend using a model with the higher accuracy score and lower computational costs. The accuracy for random forest and neural network are almost same so it all comes down to the computational cost and time required to build the model.

The training time and computational resources required to train a random forest or neural network can vary depending on the size of the dataset and the complexity of the model. In general, random forests are faster to train and require less computational resources than neural networks, especially for large datasets. So, I would recommend the school to use the random forest model to check if students need to work on their motor skills at a young age.

Appendix


label
pixel43
pixel44
pixel92
pixel124
pixel125
pixel126
pixel127
pixel128
pixel129
pixel130
pixel131
pixel132
pixel133
pixel134
pixel135
pixel136
pixel137
pixel138
pixel146
pixel147
pixel148
pixel149
pixel150
pixel151
pixel152
pixel153
pixel154
pixel155
pixel156
pixel157
pixel158
pixel159
pixel160
pixel327
pixel328
pixel329
pixel351
pixel410
pixel411
pixel412
pixel413
pixel414
pixel415
pixel416
pixel417
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



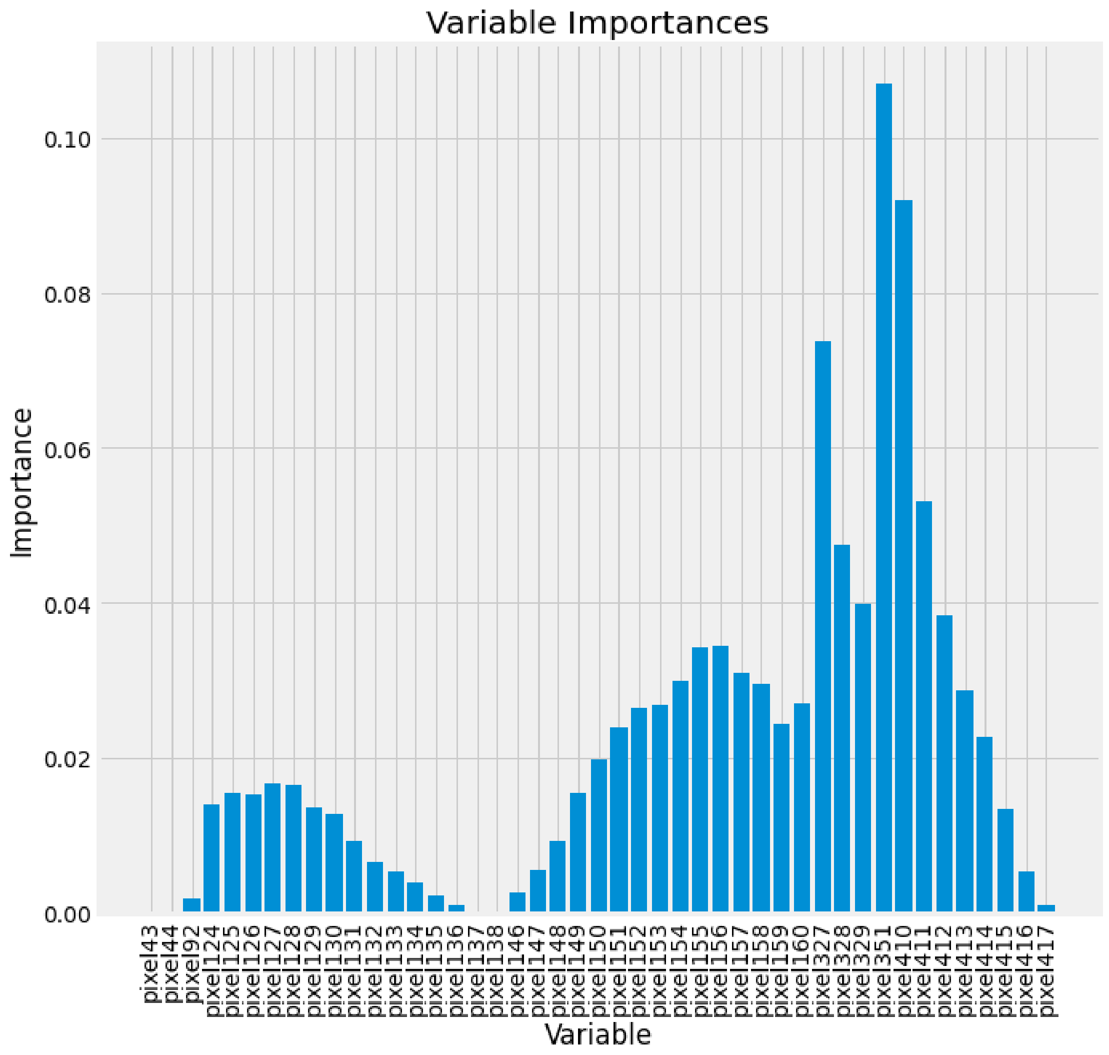
	Statistics
Min	0.000000
Q1	2.000000
Median	4.000000
Q3	7.000000
Max	9.000000
Mean	4.390542
StdDev	2.897530

	Value counts
Positive values	36235
Zero values	4132
Negative values	0

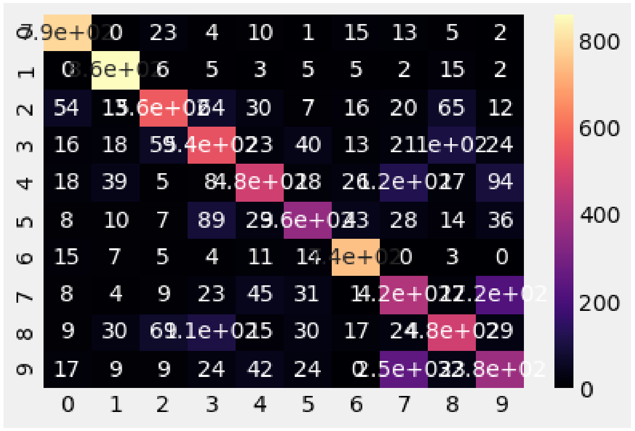
Min: 0
Bin width: 1
Max: 9.5

```
              precision    recall  f1-score   support

           0       0.84      0.90      0.87       864
           1       0.79      0.96      0.86       902
           2       0.68      0.63      0.65       840
           3       0.57      0.60      0.59       856
           4       0.69      0.50      0.58       823
           5       0.62      0.48      0.54       621
           6       0.84      0.90      0.87       804
           7       0.47      0.65      0.54       772
           8       0.64      0.51      0.57       811
           9       0.47      0.43      0.45       781

    accuracy                           0.66      8074
   macro avg       0.66      0.66      0.65      8074
weighted avg       0.66      0.66      0.66      8074

0.6644785731979193
```

```
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       864
           1       0.87      0.95      0.91       902
           2       0.74      0.67      0.70       840
           3       0.62      0.63      0.63       856
           4       0.70      0.58      0.64       823
           5       0.68      0.57      0.62       621
           6       0.85      0.93      0.88       804
           7       0.47      0.54      0.50       772
           8       0.64      0.59      0.62       811
           9       0.47      0.48      0.48       781

    accuracy                           0.69      8074
   macro avg       0.69      0.69      0.69      8074
weighted avg       0.69      0.69      0.69      8074

0.6940797621996532
```



```
Confusion matrix:
 [[791   0  23   4  10   1  15  13   5   2]
 [  0 859   6   5   3   5   5   2  15   2]
 [ 54  13 559  64  30   7  16  20  65  12]
 [ 16  18  59 539  23  40  13  21 103  24]
 [ 18  39   5   8 480  18  26 118  17  94]
 [  8  10   7  89  29 357  43  28  14  36]
 [ 15   7   5   4  11  14 745   0   3   0]
 [  8   4   9  23  45  31   1 417  17 217]
 [  9  30  69 107  15  30  17  24 481  29]
 [ 17   9   9  24  42  24   0 248  32 376]]
```



```
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       864
           1       0.87      0.95      0.91       902
           2       0.74      0.67      0.70       840
           3       0.62      0.63      0.63       856
           4       0.70      0.58      0.64       823
           5       0.68      0.57      0.62       621
           6       0.85      0.93      0.88       804
           7       0.47      0.54      0.50       772
           8       0.64      0.59      0.62       811
           9       0.47      0.48      0.48       781

    accuracy                           0.69      8074
   macro avg       0.69      0.69      0.69      8074
weighted avg       0.69      0.69      0.69      8074

0.6940797621996532
```