

**Module 5 Project**

**ALY 6020**

Name: Kavitha Palthayvath

Instructor’s name: Justin Grosz

Due Date: 08/15/2021

**Abstract**

Inculcating personal development in children is necessary at an early age. Evaluating and improving motor skills has become an integral part of the personal development in children. Finding if a student needs help in improving motor skills is a big problem and cannot be relied on instincts of teachers directly. In this paper we are focusing on using handwriting as a method to point out if a student may need to work on their motor skills. We are going to feed the pixel level intensity of the handwriting of the student and see if the model we train can accurately predict if it can accurately understand and relate to the correct number.

**Introduction**

The dataset provided to us has details about pixel level intensity of the handwriting of student and we are using this data to train machine learning models to accurately predict the letter which the handwriting represents. To use the given data most usefully we performed a through data analysis to train different models and compare their performance metrics to find the best suitable model.

**Analysis**

**Data Cleaning**

This dataset did not require any data cleaning.

**Model Comparison and Analysis**

***K Nearest Neighbor***

We started with training a KNN model for predicting the numbers accurately from handwriting. KNN is well known for classification of labelled data to produce an output for unseen data. It uses the Euclidean distance formula to compute the distance between the data points for prediction. We trained the KNN model for different values of ‘K’ and calculated the score/accuracy of each model to find the right value of ‘K’ for which the model performs the best. As per Figure 1, the best value of ‘K’ came out to be 2. The benchmark metrics for this model came out as, 60.55% accuracy, Mean Absolute error as 1.307 and took 4.029 seconds to execute. Though this model does give 60% accuracy, there is no scope of improving this accuracy by changing any parameters for this model. KNN is known to be a ‘Lazy Learner’, it just gets the data and not learn from it, it does not use any discriminative function to find unique features in variables to have better predictive power. A specific problem for using KNN for this dataset is that this dataset is a large dataset and it is recommended to use KNN for small dataset because KNN is sensitive to noise present in data, and large datasets have higher amount of noise. KNN is also not recommended for datasets with higher dimensions (high number of columns, this dataset has 46 columns), because more the dimensions more time needed to calculate distance. Last but not least KNN does more computation in the test time rather than while training, so regardless or being trained with good dataset it will take more time in prediction.

***Random Forest***

Next, we trained a random forest classifier model, random forest algorithm is very well known for solving classification problems with good accuracy. We trained the model without any feature selection and the benchmark metrics of this model gave us, 98.54% accuracy, mean absolute error of 0.0234, and executed in 4.54 seconds. We then decided to attempt making the model better by using feature selection. Using only important features can give a good performance boost. On using feature selection and retraining the model the benchmark metrics we got were as follows, accuracy 70.07%, mean absolute error of 0.97 and execution time of 6.15 seconds. This only made things worse, this was a big problem for us with this dataset hence we decided to stick with using all features. Next, we thought of using a greater number of trees in the random forest while considering all features to see if we can remove any overfitting if there is some and make the model even better. On retraining the model with 500 trees, the benchmark metrics we got were, 98.41% accuracy, mean absolute error of 0.022 and an execution time of 22.50. This model in comparison with the model with all features and lesser trees had 0.1% lesser accuracy but had lesser error by 0.001, but the biggest problem with this model is that it took 22 seconds to execute which is 5 times the time the first random forest model took.

***Neural Network***

In search of the best model, we then decided to use Neural Network, Multi-layer Perceptron classifier to be specific. This model is known for providing better classification prediction using ‘n’ layers of perceptron which have ‘m’ number of neurons each. These neurons have predictive power of classifying data. We trained a model with 3 perceptron layers with 13 neurons in each layer in batches of 500 records. The benchmark metrics of this mode gave us, 99.84% accuracy, 0.0019 mean absolute error and execution time of 29.25 seconds. This model clearly outperformed the decision tree but still there is the question of execution time. To make the model better by removing overfitting we optimized the number of neurons in each perceptron and chose 8 neurons in each layer and retrained the mode, the benchmark metrics of this model gave us, 99.83% accuracy with mean absolute error of 0.0017 and execution time of 29.66 seconds. Though this model did not get accuracy boost but it removed over fitting which is reflected in the error of this model which is lesser than the previous model. Finally, we tried to work on the accuracy of this model by including more layers(perceptron), hence we introduced another layer of perceptron and re trained the model with optimal number of neurons. The benchmark metrics of this model gave us, 99.97% accuracy, with 0.00023 mean absolute error and took 32.57 seconds to execute. This is the most accurate model we trained so far.

**Benchmark Comparison**

To summarize, as per our analysis the various models we trained performed as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy** | **MAE** | **Execution**  **Time (sec)** |
| KNN | 60.55% | 1.307 | 4.02 |
| Random Forest with 100 trees and no feature selection | 98.54% | 0.0234 | 4.54 |
| Random Forest with 100 trees and feature selection | 70.07% | 0.9747 | 6.15 |
| Random Forest with 500 trees and no feature selection | 98.41% | 0.022 | 22.50 |
| Neural Network | 99.84% | 0.0019 | 29.25 |
| Neural Network with optimal number of neurons | 99.83% | 0.0017 | 29.66 |
| Neural Network with optimal number of neurons and layers | 99.97% | 0.0002 | 32.57 |

**Conclusion**

On comparing all the models, we would recommend to use the Neural Network model with optimal number of neuron and layers because it provides 99.97% accuracy with barely any error. Though this model takes 32.57 seconds to execute which is the maximum time any model takes to execute but to address the problem statement we are supposed to make least errors as possible and time taken to predict the result is not of as much importance. Though someone can make a good argument of using the random forest model with 100 tress and no feature selection as it has good accuracy, some error but executes in 4 seconds but as per the problem statement there is no mention of making the prediction faster and we do not want speed at the cost of being inaccurate. As the result of relying on a model which does not have good accuracy would be to wrongly recommend a child to work on his/her motor skills. To solve this business problem we need to choose a model which has the best accuracy, and least error hence choosing the Neural Network model with optimal number of neuron and layers is the best choice.

**References**

1. *Beginners Ask “How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?”* . (2018, June). Retrieved from towards datascience: https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e
2. Portilla, J. (2017, March). *A Beginner’s Guide to Neural Networks in Python*. Retrieved from springboard: https://www.springboard.com/blog/ai-machine-learning/beginners-guide-neural-network-in-python-scikit-learn-0-18/

**Appendix**

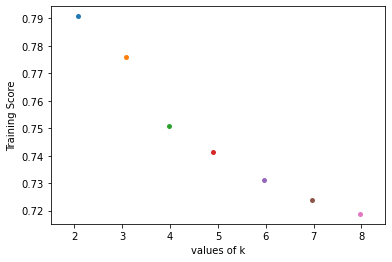
****

Fig 1 : Training Score vs Values of K