**ELC Handwritten Digit Recognition System**

We used the KNN Classifier algorithm to recognise handwritten Digits.

**About KNN Classifier:**

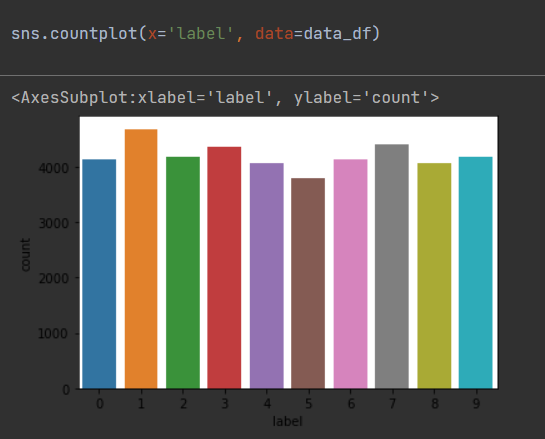
K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning. KNN assumes the similarity between new case and available cases and puts the new case into the category that is most similar to a particular category. It is a non-parametric algorithm i.e.; it does not make any assumption on the underlying data. It is also famously known as the lazy learner algorithm as it does not learn from the training set immediately, instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN has the advantage that it is easy to implement. The larger the data available, the better is the prediction accuracy. The main disadvantage of KNN algorithm is that it’s computation cost is high because of calculating the distance between the data points for all of the thousands of the training samples.**[1]**

The process I used included 3 main steps which are described below:

**STEP 1: Check the given dataset**

So in the first part of the program we checked our data set for any inaccuracies and imbalanced nature. For a good model we aim to have a well-balanced data set. From Well-balanced we mean that enough number of data is available for each possible conclusion. In this case we were checking if there are enough samples for all the digits and the amount of data available for each digit is roughly the same. We also checked the data set of certain points and correlated if the image and the corresponding conclusion (i.e., the deduction in the provided data-set) is the same.

Given below is the screenshot showing that our data is well-balanced:



**STEP 2: Train our model**

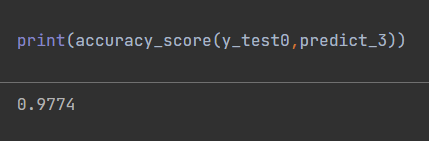
For this purpose, we made use of some invaluable python libraries. To implement KNN we imported the KNeighborsClassifier library. For further checking of the data, we also imported accuracy\_score, classification\_report and confusion matrix to display all the data required to determine if our model is working well. After importing we passed the test.csv values into our model.

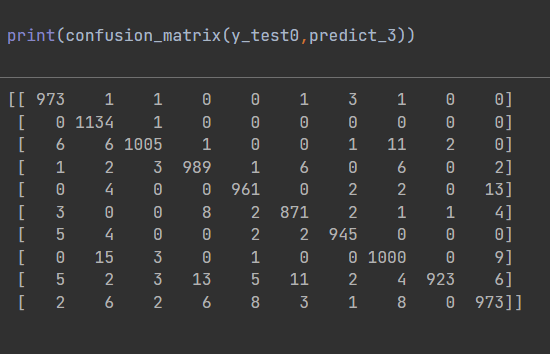
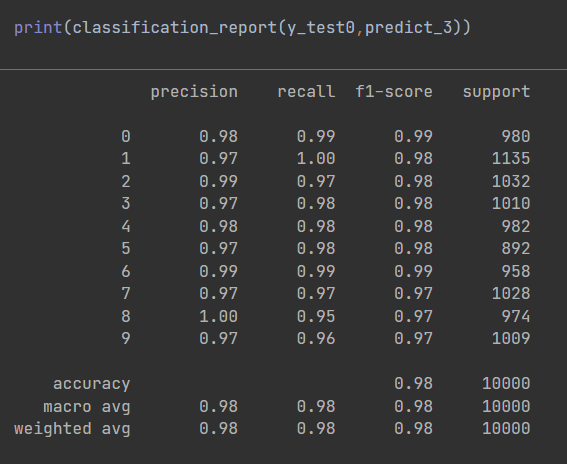
**STEP 3: Finding the ideal K value**

For the KNN classifier we have the freedom to choose any value of K we want. But to find that out we have to do some trial and error. The value of K chosen is basically the number of points whose Elucidan distance will be calculated with respect to a given point in the data-set being judged. Hence, I decided to check my model using K=3,9,2,4,5 and 13. Each of these values of K gave different accuracies. From our results we determined that the result obtained with K=3 was the best and hence while using this model I would suggest the user to use K=3 for their prediction.

**FINAL RESULTS: K=3 Accuracy=97.74%**

Given below are the screenshots showing the accuracy, classifier report and confusion matrix for the best value of K i.e., K=3.





**References:**

1. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>