# **KNN Classification Model on MNIST data**

Krishna Kumar Veeraputhiran

Grand Canyon University

DSC 540 – O500 – Machine Learning for Data Science

Dr. Aiman Darwiche

August 18,2021

# **KNN Classification Model on MNIST data**

This Technical report provides details on how K- Nearest Neighbor (KNN) classification algorithm has been used to predict the value labels for MNIST (Modified National Institute of Standards and Technology) database for handwritten digits. We will discuss on steps like how the training dataset and the test dataset have been preprocessed before a model is fitted to predict the test data. We will also discuss on various Model evaluation metrics for the built KNN model to evaluate the performance of the final model.

**About the Data**

The MNIST database for handwritten digits has 4 datasets, two training files with one having 60,000 images of digits (from 0 to 9) which are of dimension 28 X 28 and the other file has the corresponding label for the images. The test set similarly has 10,000 images and labels in two files. In order to train a model on the images to predict the corresponding labels, they are pre-processed to an array of dimension (60000, 28\*28) with each image represented as a row in the array with 28\*28 columns. The test image file is converted to an array of dimension (10000, 28\*28)

**KNN Model**

According to Gopal (2019), the KNN algorithm is a supervised ML classification algorithm which is data-driven and does not make any assumptions about the model between the predictor and the predicted variable. This method extracts the similarity pattern existing among the training dataset and uses it to classify the test dataset.

The Algorithm of the KNN model’s algorithm can be given as follows: a) Load the data b) Initialize K to the chosen number of neighbors c) For the test data, calculate the distance between the training data and the test data. Once calculate the distance and the index is added to an ordered collection like an array d) The ordered collection is sorted in ascending based on the distance e) The first k entries are selected f) Out of the k entries the label with most number gets the highest vote and the data point is classified to the corresponding label.

In our model the training dataset is broken into two sets namely training (80%) and validation set (20%). This is to calculate the accuracy of the KNN model at different k –values. The KNN model is fitted using the python’s scikit-learn library function called KneighborsClassifier. The 80% of the training dataset is first used to train a fitted model. Then this model is used on the validation set (20% of training set) to calculated the accuracy. The model accuracy is calculated for values of K ranging from 1 to 9. It is noted that the model’s accuracy is highest for k=3 (accuracy at 97.5%) and the accuracy starts to drop when k value is increased beyond 3. Hence, we select our k value to be 3 to fit our test dataset.

In the model to fit the training dataset, as decided we use the k value as 3 and use the Euclidean distance to calculate the nearest points (p=2) for classification. Once the model is created, we use the predict function of the fitted model to predict the label classification of the test dataset. By running the classification\_report() function on the predicted outcome vs the original test label we can find the overall accuracy of the model (which is calculated at 97%) and the precision (Ratio of True Positive and Total Predicted Positive) and of each label being classified. From the report we can find that the precision of predicting the digit 8 based on the image input is the highest (99%) followed by digits 6 and 2 (96%).

**Euclidean Distance**

Euclidean distance is one of the distance metric which is a type of Data similarity measure to classify data based on distance. As mentioned earlier the KNN algorithm that we used to classify the test digits are grouped based on the Euclidean distance with the k nearest neighbors formed by the training dataset. The Euclidean distance dil , between two pints i and l according to the Gopal (2019) can be given as,

In our model the Euclidean distance and the k nearest neighbors can be calculated by one of methods of the knn model namely ‘kneighbors‘. This method when applied on the test data, will provide the corresponding k neighbors and the Euclidean distance between a data point in the test dataset to it’s corresponding k neighbors obtained from the training dataset. In our model we can see that the most popular neighbor when k =1 is 1, with 1221 votes and the least popular neighbor with k=1 is 5 with 893 votes. Also the most predicted value from the test dataset is 1 (with 1181 predictions) even though there are only 1135 images which represent 1 in the test dataset.

**Confusion Matrix**

Confusion matrix is one of the model evaluation tool used to evaluate the performance of a classification model. In our case we have used the confusion matrix to predict accuracy of the digits classified based. The confusion matrix for our model is built using the plot\_confusion\_matrix method from the scikit-learn package. The y-axis of the confusion matrix shows the original label values while the x-axis provides the value predicted by the model. The confusion matrix shows that the accuracy of each labels with 1 being the with 1132 data points correctly predicted out of the 1135. The diagonal element in the confusion matrix provides the total number of the data points that are correctly predicted. One of the biggest advantage of the confusion matrix in a classification model is that, we can predict the accuracy of each and every individual classes and this provides great insight on which classes may need additional features to predict the values right. From our confusion matrix we can see that the accuracy of all the labels are over 95%. The accuracy of any given labels can be calculated by finding the ratio between the correctly predict number of data (diagonal cell) to that of the total number of actual data for the given label (sum of the horizontal row for a label)

**The ROC Curve**

The Receiver Operator Curve (ROC) is an evaluation metric commonly used to evaluate binary classification model. In our model we have calculated the AUC (Area under the curve) for the overall model by using the predicted probability of the test data. By default ROC\_AUC will be calculated only for binary classifiers. In our case we have used an additional option in the function roc\_auc\_score called multi\_class and set the value as ‘ovr’ (one-vs-rest). In this was the auc of each class is computed against other class. We see that the AUC for the model is close to 1 (0.9924) which means the model is able to efficiently distinguish between the positive and negative classes accurately. The individual ROC-AUC is also calculated for each class by having the positive predicted probability of the test dataset as input. We can see all the labels have an AUC greater than 0.985.

Conclusion

This work is carried out to get a better understanding of the k- Nearest Neighbor classification algorithm and the MNIST imaged based digits dataset. In this report we discussed about the KNN model applied on MNIST digit images and portrayed our findings on the overall effectiveness of the model using different evaluation methods. The overall accuracy of the model is at 97% and the AUC for the overall model is at 0.9924, meaning the model is performing great in distinguishing between the positive and negative classes accurately. Also the error rate of the model is at the lowest for k=3 (at 3.12%) compared to other values of k.

# **References**

Larose, C. D., & Larose, D. T. (2019). *Data science using Python and R*. John Wiley & Sons, Inc.

Gopal, M. (2019). *Applied machine learning*. McGraw-Hill Education.

U. R. Babu, Y. Venkateswarlu and A. K. Chintha, "Handwritten Digit Recognition Using K-Nearest Neighbour Classifier," 2014 World Congress on Computing and Communication Technologies, 2014, pp. 60-65, doi: 10.1109/WCCCT.2014.7.

YouTube. (2017). *K-Nearest Neighbor Classification Mnist Handwritten Digits*. *YouTube*. https://www.youtube.com/watch?v=ooQtUaCExa8.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Adya, S. (2019, May 16). *K-Nn for mnist classification*. Back-Prop. https://www.back-prop.com/deep\_learning/knn/mnist/2019/05/16/knn\_classifier/.

Brownlee, J. (2018, August 31). *How to USE Roc curves and PRECISION-RECALL curves for classification in Python*. Machine Learning Mastery. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.