HANDWRITTEN DIGIT CLASSIFICATION

Abstract

Pattern recognition is one of the major challenges in the statistics framework. Its goal is the feature extraction to classify the patterns into categories. A well-known example in this field is the handwritten digit recognition where digits have to be assigned into one of the 10 classes using some classification method. Our purpose is to present alternative
classification methods based on statistical techniques. Experiments are performed on the known MNIST databases.

Contents

List of Figures

Abbreviations and Acronyms

CNN	**Convolutional Neural Network**
K-NN	**k nearest neighbor**
LOOCV	**Leave-One-Out Cross-Validation**

1. **Introduction**

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. Neighbors-based methods are known as *non-generalizing* machine learning methods, since they simply "remember" all of its training data.

1.1. Problem Statement

The task at hand is to classify handwritten digits using supervised machine learning methods. The digits belong to classes of 0 – 9. "Given a query instance (a digit) in the form of an image, our machine learning model must correctly classify its appropriate class."

1.2. Research Questions

We break down the problem and summarise it using 3 research questions that we attempt to answer in this thesis.

1. How can Implement a K-Nearest Neighbor (KNN) classifier and apply the model to a data and get an acceptable accuracy.

2. How can we use the leave-one-out cross validation approach for determining the best K value.

3. How can we construct a confusion matrix showing the number of images of the Test folder of each digit that were classified to belong to different digits.

1.3. Related Work

Handwritten digits are a common part of everyday life. These days machine learning methods are classifying handwritten digits with accuracy exceeding human accuracy. These methods are used to make Optical Character Recognizers (OCRs) which involve reading text from paper and translating the images into a form that the computer can manipulate (for example, into ASCII codes). This technology is solving a plethora of problems like automated recognition of:

1- Zip Codes:

A zip code consists of some digits and is one of the most important parts of a letter for it to be delivered to the correct location. Many years ago, the postman would read the zip code manually for delivery. However, this type of work is now automated by using optical character recognition (OCR).

2- Bank Cheques:

The most frequent use of OCR is to handle cheques: a handwritten cheque is scanned, its contents converted into digital text, the signature verified and the cheque cleared in real time, all without human involvement.

3- Legal Records:

Reams and reams of affidavits, judgements, filings, statements, wills and other legal documents, especially the printed ones, can be digitized, stored,

databased and made searchable using the simplest of OCR readers.

4- Healthcare Records:

Having one's entire medical history on a searchable, digital store means that things like past illnesses and treatments, diagnostic tests, hospital records, insurance payments etc. can be made available in one unified place, rather than having to maintain unwieldy files of reports and X-rays. OCR can be used to scan these documents and create a digital database.

2. Software System Tools

2.1. Numpy

Is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The core functionality of NumPy is its "ndarray", for *n*-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure, these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be viewed into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably

BLAS and LAPACK). NumPy has built-in support for memory-mapped ndarrays.

2.2. Anaconda

Anaconda is a Python and R distribution. It aims to provide everything needed for data science tasks, Anaconda uses Conda as the package manager.

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. It quickly installs, runs and updates packages and their dependencies. It also easily creates, saves, loads and switches between environments on your local computer. - It was created for Python programs, but it can package and distribute software for any language. - Conda is written entirely in Python which makes it easier to use in Python virtual environments. Furthermore, we can use Conda for C libraries, R packages, Java packages and so on.

2.3. Pandas

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license . The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

2.4. Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, since then it has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

2.5. Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

2.5. Sklearn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn is largely written in Python, and uses numpy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as matplotlib and plotly for plotting, numpy for array vectorization, pandas dataframes, scipy, and many more.

3. System Architecture

3.1. Module 1: Datasets

MNIST Handwritten Digits dataset is used for this task. It contains images of digits taken from a variety of scanned documents, normalized in size and centered. This makes it an excellent dataset for evaluating models, allowing the developer to focus on machine learning with very little data cleaning or preparation required. Each image is a 28 by 28 pixel square (784 pixels total). The dataset contains 60,000 images for model training and 10,000 images for the evaluation of the model.

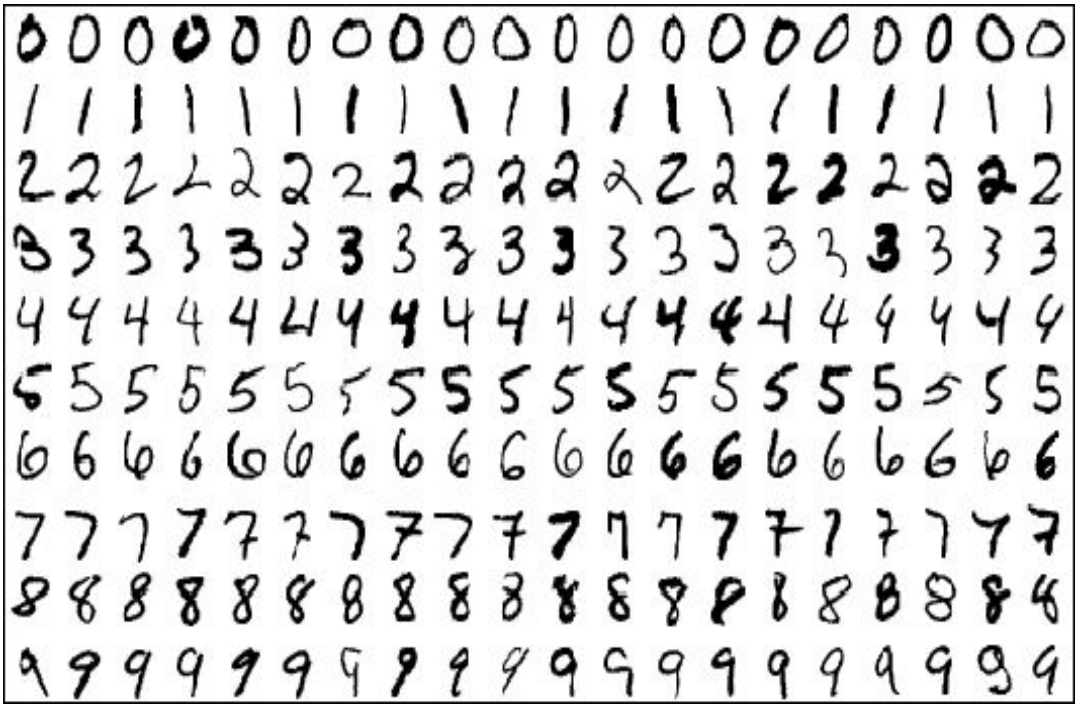

Figure 1: Real MNIST Handwritten Digits

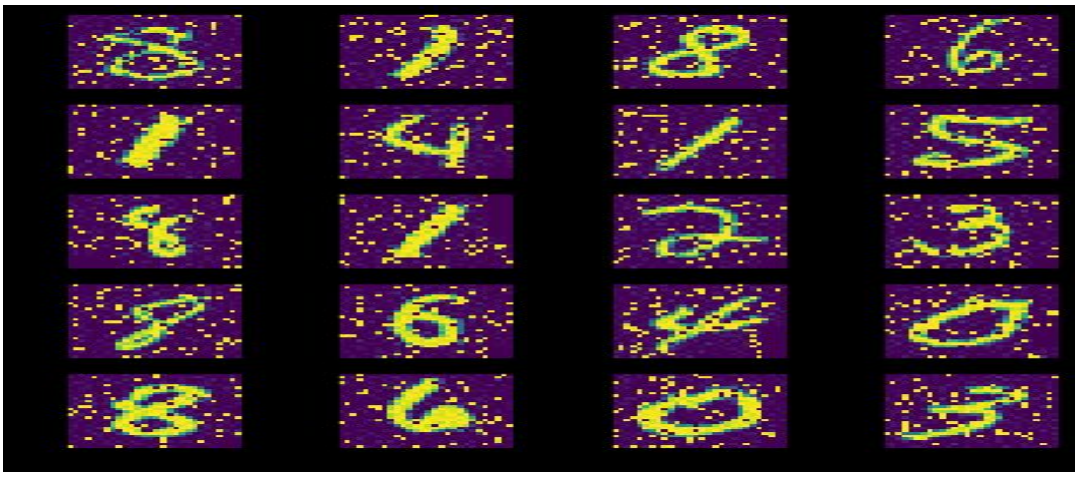

Figure 2: MNIST Handwritten Digits After Importing

In our task we use a part of Mnist dataset we had two files consists of :

1) Training data : 2400 images for handwritten digits packed into 10 packs each pack has 240 images.

2) Testing data : 200 images for handwritten digits packed into 10 packs each pack has 20 images.

The Training datasets were homogeneous and packed into 10 packs each pack has 240 images.

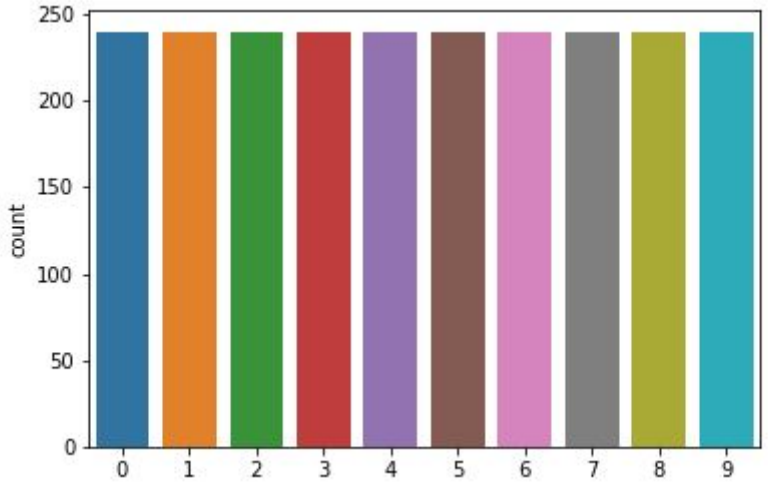

Figure 3: Homogeneous Training & Testing Datasets

3.2. k-Nearest Neighbors

3.2.2. Overview

The k-Nearest Neighbors algorithm or KNN for short is a very simple technique.

The entire training dataset is stored. When a prediction is required, the k-most similar records to a new record from the training dataset are then located. From these neighbors, a summarized prediction is made.

Similarity between records can be measured many different ways. A problem or data-specific method can be used. Generally, with tabular data, a good starting point is the Euclidean distance.

Once the neighbors are discovered, the summary prediction can be made by returning the most common outcome or taking the average. As such, KNN can be used for classification or regression problems.

There is no model to speak of other than holding the entire training dataset. Because no work is done until a prediction is required, KNN is often referred to as a lazy learning method.

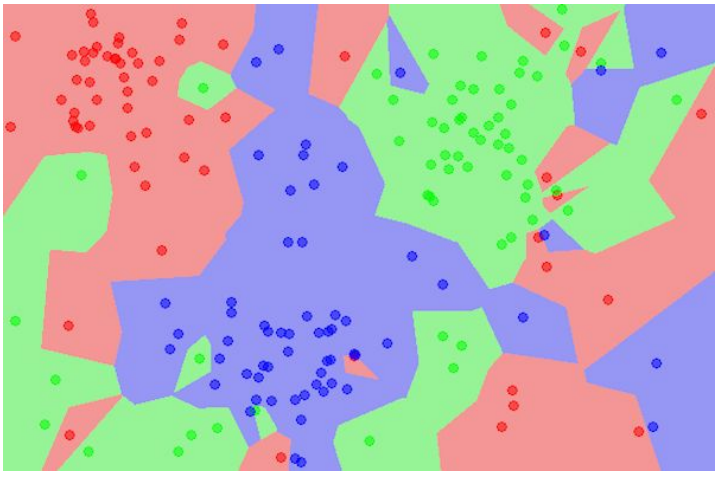

Figure 4: Effect of K-NN to datasets

3.2.2. Calculate Euclidean Distance

The first step is to calculate the distance between two rows in a dataset. Rows of data are mostly made up of numbers and an easy way to calculate the distance between two rows or vectors of numbers is to draw a straight line. This makes sense in 2D or 3D and scales nicely to higher dimensions.

We can calculate the straight line distance between two images using the Euclidean distance measure. It is calculated as the square root of the sum of the squared differences between the two images.

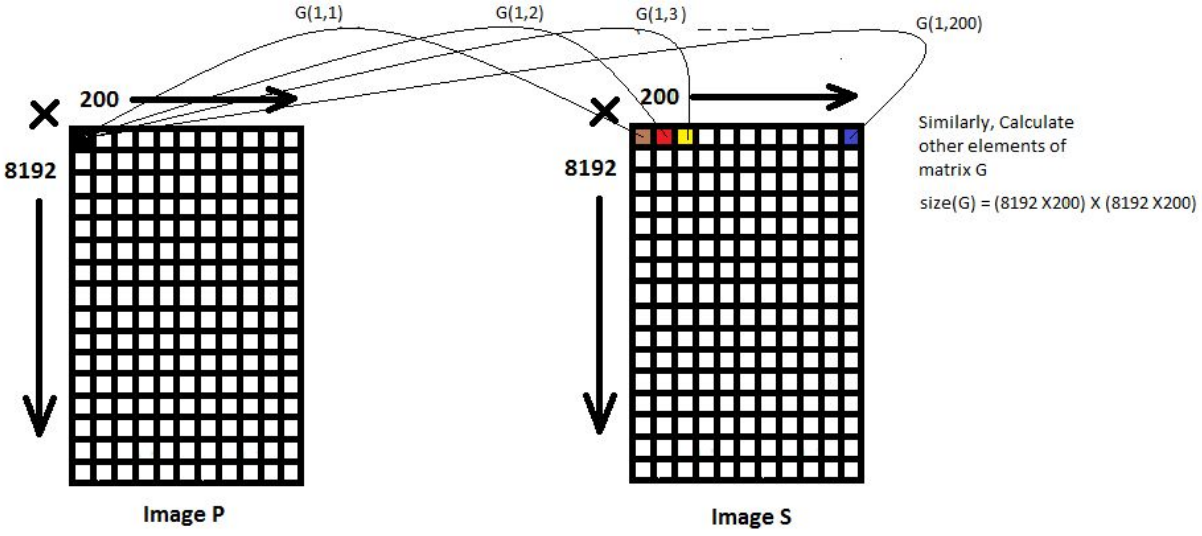

Figure 5: Euclidean Distance Between Two Images

3.2.3. Get Nearest Neighbors

Neighbors for a new piece of data in the dataset are the *k* closest instances, as defined by our distance measure.

To locate the neighbors for a new piece of data within a dataset we must first calculate the distance between each record in the dataset to the new piece of data. Once distances are calculated, we must sort all of the records in the training dataset by their distance to the new data. We can then select the top *k* to return as the most similar neighbors.

3.2.4. Make Predictions

The most similar neighbors collected from the training dataset can be used to make predictions.

In the case of classification, we can return the most represented class among the neighbors.

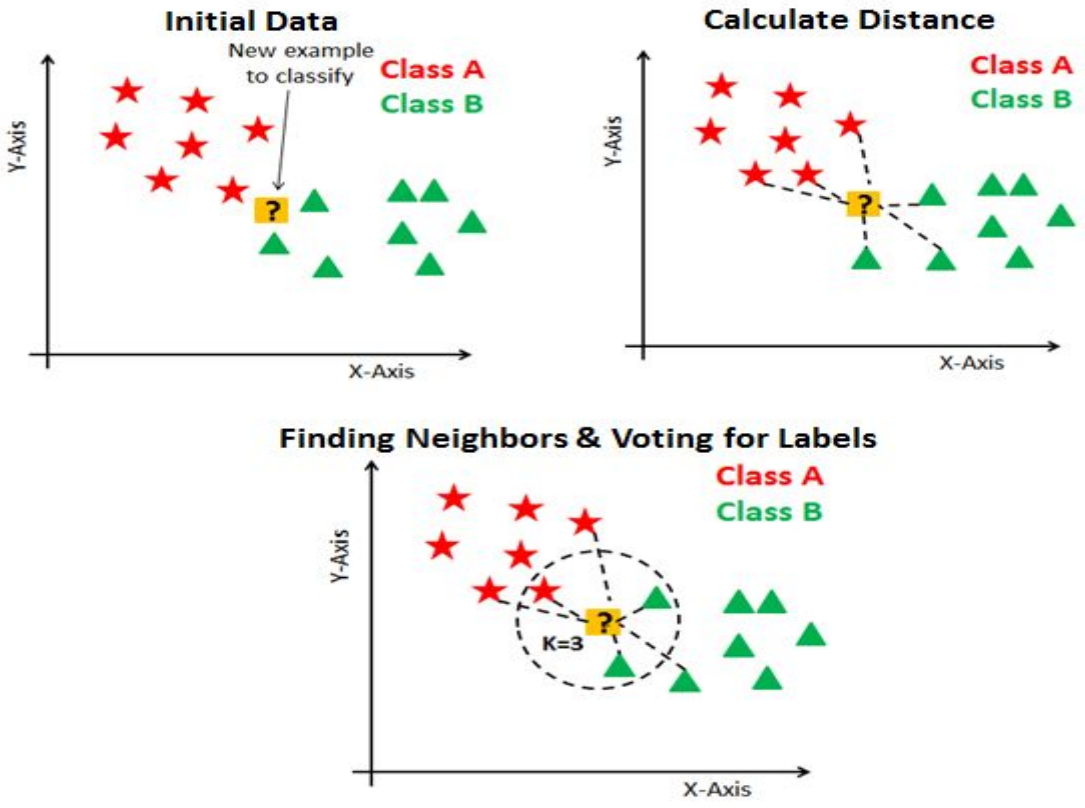

Figure 6: Steps Of K-NN Algorithm

3.3. Leave-one-out cross-validation

3.3.1. Overview

The Leave-One-Out Cross-Validation, or LOOCV, procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a computationally expensive procedure to perform, although it results in a reliable and unbiased estimate of model performance. Although simple to use and no configuration to specify, there are times when the procedure should not be used, such as when you have a very large dataset or a computationally expensive model to evaluate.

The cross-validation has a single hyperparameter "*k*" that controls the number of subsets that a dataset is split into. Once split, each subset is given the opportunity to be used as a test set while all other subsets together are used as a training dataset.

This means that k-fold cross-validation involves fitting and evaluating *k* models. This, in turn, provides k estimates of a model's performance on the dataset, which can be reported using summary statistics such as the mean and standard deviation. This score can then be used to compare and ultimately select a model and configuration to use as the "*final model*" for a dataset.

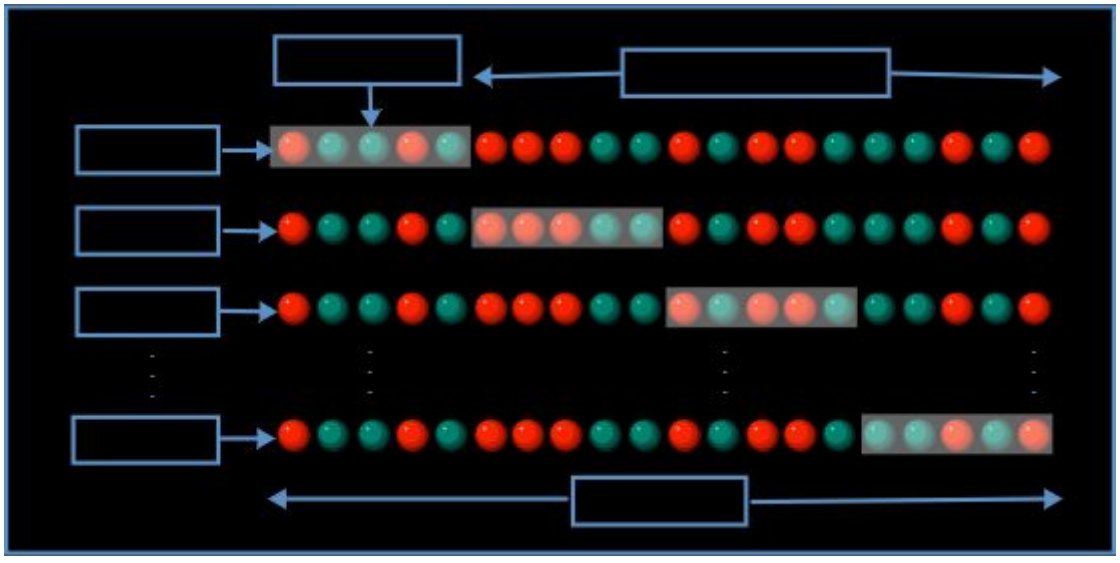

Figure 7: LOOCV Process

3.3.2. Choosing Best-K

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

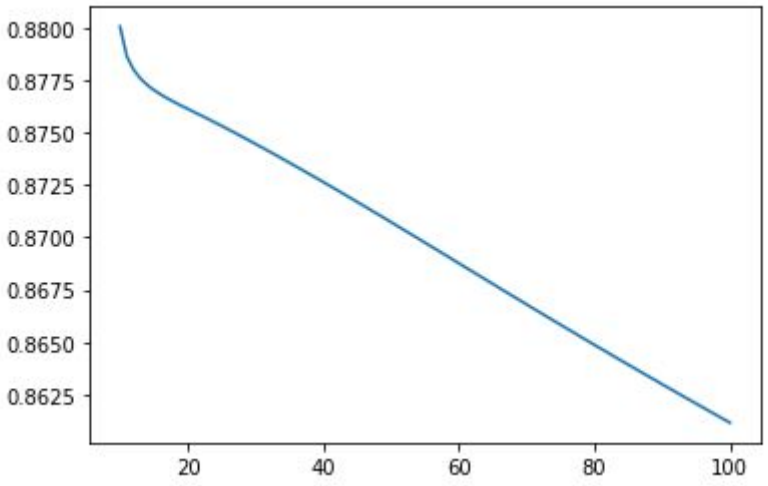

Figure 8: Best-K Plot

4. Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

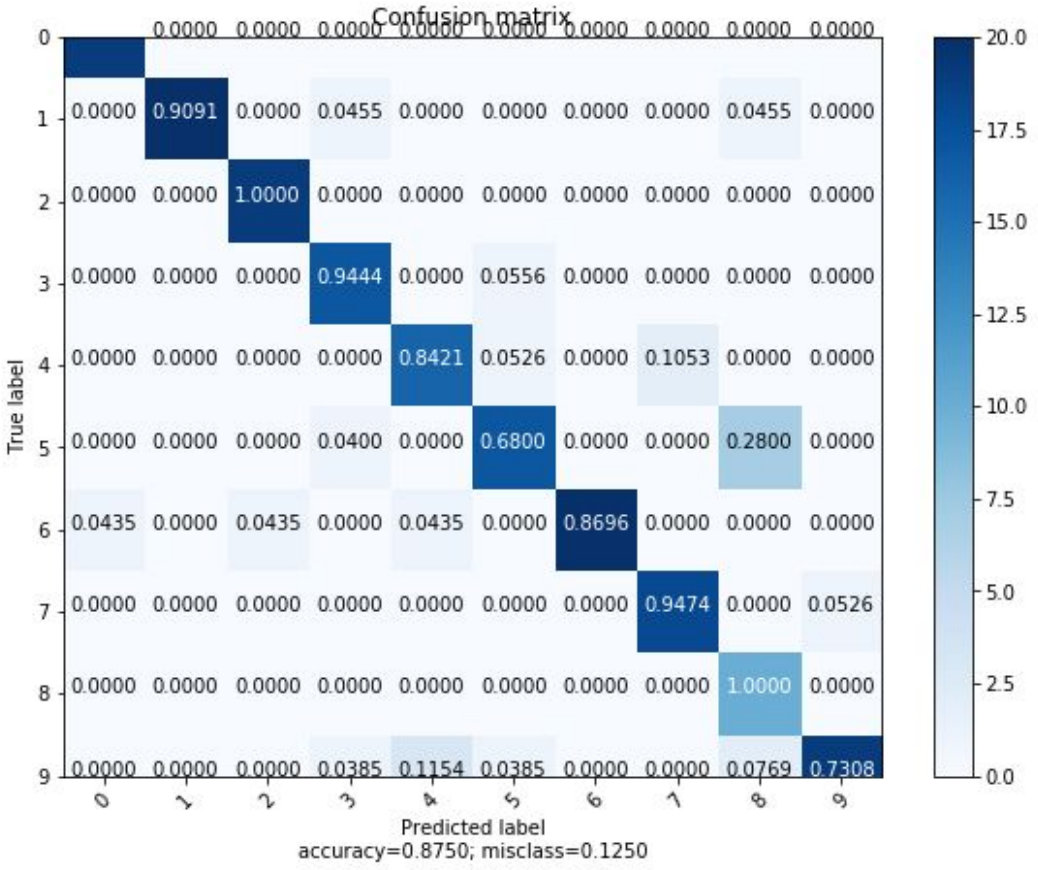

Figure 9: Confusion Matrix

5. Results

After training the k nearest neighbor (K-NN) with 2400 images and using the Leave-One-Out Cross-Validation (LOOCV) to validate the model we get :

Testing accuracy by test 200 image equals 87.5%

Validation accuracy equals 88%

6. Conclusion

Using k nearest neighbor (K-NN) to solve this problem is a good way but not the most efficient way, we could use a simple conventional neural network (CNN) to solve such a problem.

Using a CNN in this problem would be efficient as K-NN consumes huge memory space especially in case of huge datasets while CNN did not suffer from these problems.

The leave-one-out cross-validation procedure is appropriate when you have a small dataset like : Mnist datasets or when an accurate estimate of model performance is more important than the computational cost of the method.

We can increase the accuracy of the model if we choose another classifier or if we train the model with full mnist datasets as we only use 2400 images for training and 200 image for testing.