SPL-1 Project Report, 2019

Hand Written Digit Recognition Software

Course: Software Project Lab I

Course No: SE 305

Submitted by

Fahim Ahmed

Roll No. : BSSE 1039

Session: 2016-2017

Supervised by

Dr. Naushin Nower

Designation: Associate Professor

Institute of Information Technology

Institute of Information Technology

University of Dhaka

29-05-2019

Table of Contents

Index of Figures

1.Introduction

Digit Recognition is a form of Machine Learning. We human can understand a digit by just looking at them, but when we think of machine, it needs to be trained. Again, every human has different hand writings, so we cannot tell our machine exactly what a digit looks like. There are several machine learning Algorithms of different accuracies to estimate a probability of – what the hand written digit could be.

In my project my target is to extract image files into binary images and use Machine Learning Algorithms to teach my system "which digit is written".

1.1 Background study

Different image formats

Image file formats are standardized means of organizing and storing digital images.
Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer. An image file format may store data in uncompressed, compressed, or vector formats. Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.
Some popular image formats are JPEG, JPG, BMP, PNG etc.

BMP File Format

A BMP file is an uncompressed raster image comprised of a rectangular grid of pixels. It contains a file header (bitmap, identifier, file size, width, height, color options, and bitmap data starting point) and bitmap pixels, each with a different color.

The basic file structure is binary (as opposed to a text file) and is broken into the following four sections:

The File Header (14 bytes)

- Confirms that the file is (at least probably) a BMP file.
- Tells exactly how large the file is.
- Tells where the actual image data is located within the file.

The Image Header (40 bytes in the versions of interest)

- Tells how large the image is (rows and columns).
- Tells what format option is used (bits per pixel).
- Tells which type of compression, if any, is used.
- Provides other details, all of which are seldom used.

The Color Table (length varies and is not always present)

- Provides the color palette for bit depths of 8 or less.
- Provides the (optional) bit masks for bit depths of 16 and 32.
- Not used for 24-bit images.

The Pixel Data

- Pixel by pixel color information
- Row-by-row, bottom to top.
- Rows start on double word (4-byte) boundaries and are null padded if necessary.
- Each row is column-by-column, left to right.
- In 24-bit images, color order is Red, Green and Blue.
- In images less than 8-bits, the higher order bits are the left-most pixels.

Image File Reading

For reading data of an image files, I have to study some functions mentioned below

- Different types of image formats
- Which image format is suitable for me
- C/C++ functions to read BMP image
- How should I organize the data

Binary Image

A **binary image** is a digital image that has only two possible values for each pixel. Typically, the two colors used for a binary image are black and white. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

This means that each pixel is stored as a single bit—i.e., a 0 or 1. The names *black-and-white*, *B&W*, monochrome or monochromic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale image.

Naïve Bayes Classifier

Classifier:

A classifier is a machine learning model that is used to discriminate different objects based on certain features.

Principle of Naive Bayes Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

BAYES THEOREM:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Figure 1: Bayes Theorem

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

The zero-frequency problem

Add 1 to the count for every attribute value-class combination (*Laplace estimator*) when an attribute value (*Outlook=Overcast*) doesn't occur with every class value.

Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

Bernoulli Naive Bayes:

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

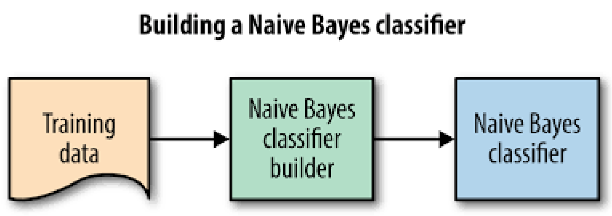

Building a Naive Bayes classifier
Training data
Naive Bayes classifier builder
Naive Bayes classifier

Figure 2: Workflow of Naïve Bayes

KNN (K Nearest Neighbors):

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure.

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

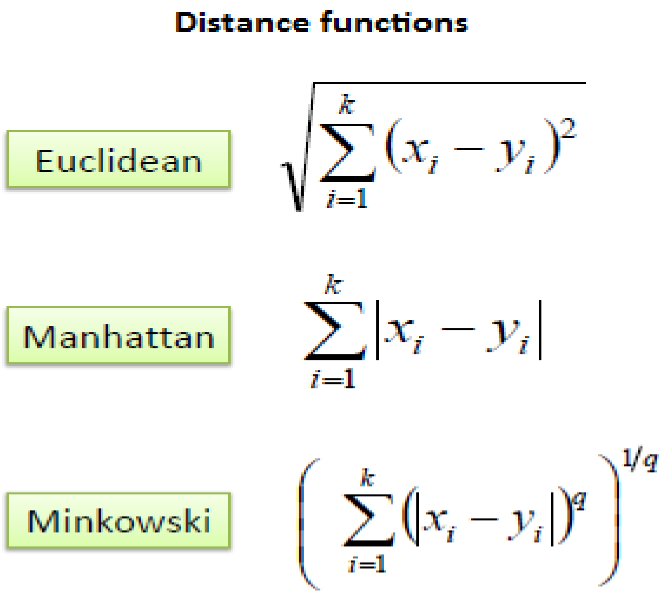

Distance functions
Euclidean
$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$
Manhattan
$\sum_{i=1}^{k}|x_i - y_i|$
Minkowski
$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Figure 3: KNN distance functions

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee.

1.2 Challenges

Implementing a software solution using algorithms carries with it a number of challenges. The process can be complex, confusing and lengthy. For implementing this project there are lots of challenges that I have faced. Some of them are

- Handling Large number of files (2000)
- Understanding Machine Learning Algorithm
- Understanding Image Files
- Reading BMP Files
- Resizing Images
- Multidimensional Arrays for manipulation and storing

2. Project Overview

I have divided my whole project into three different parts. They are

- Reading an Image and creating a binary Image
- Digit Recognition using Naïve Bayes
- Digit Recognition using KNN

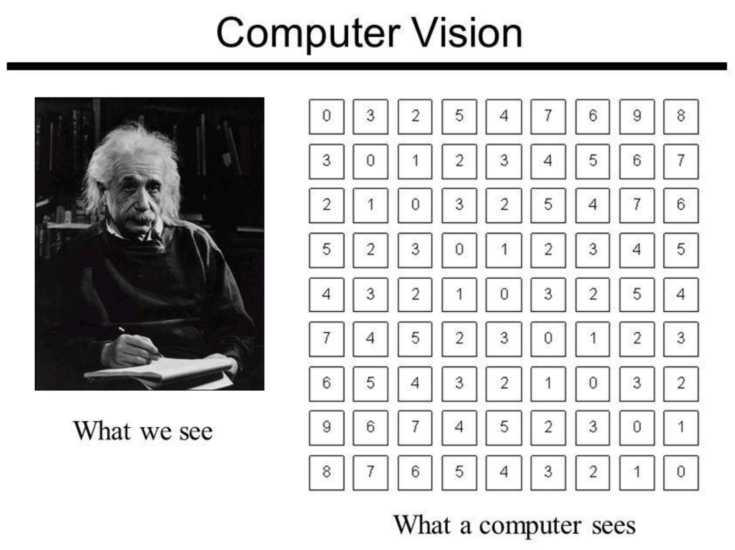
2.1. Reading and Image and creating a binary Image

This is the most important part of my project. As the main objective of my project is to detect a digit which is written on an image, at first I had to know what an image is. The way we view an image and the way a computer does are quite different. A computer actually sees an image as a pack of binary values.

But first I had to collect the data I would use. As I want my program to learn, I more data I provide, the more accurately and precisely my program will learn. For this problem, I used the MNIST dataset.

The **MNIST database** (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It has a training set of 60,000 examples, and a test set of 10,000 examples and it is written by people from various ages. So I found it quite reliable.

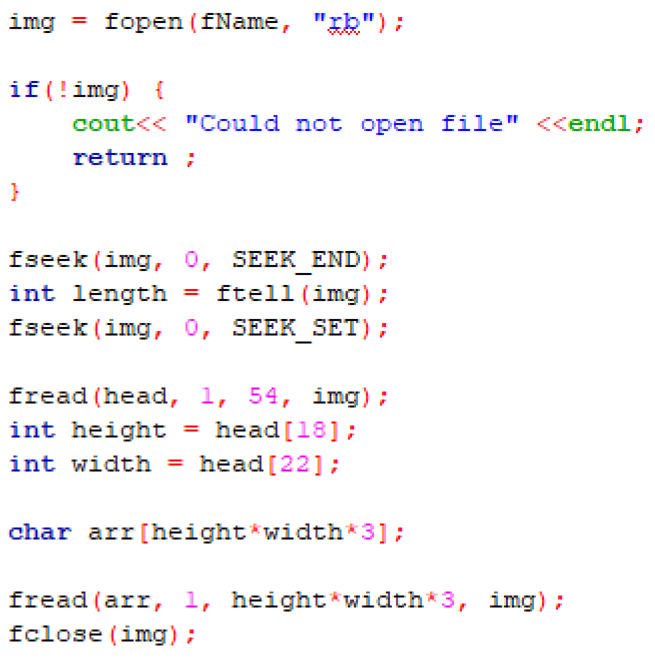
For image format, I used BMP.



0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Figure 4: Human vision vs Computer Vision

After I selected my desired dataset, than I read image. I read the file as a binary file and using file function I extracted my inputs. I took advantage of the BMP header file to find height and width of the image.

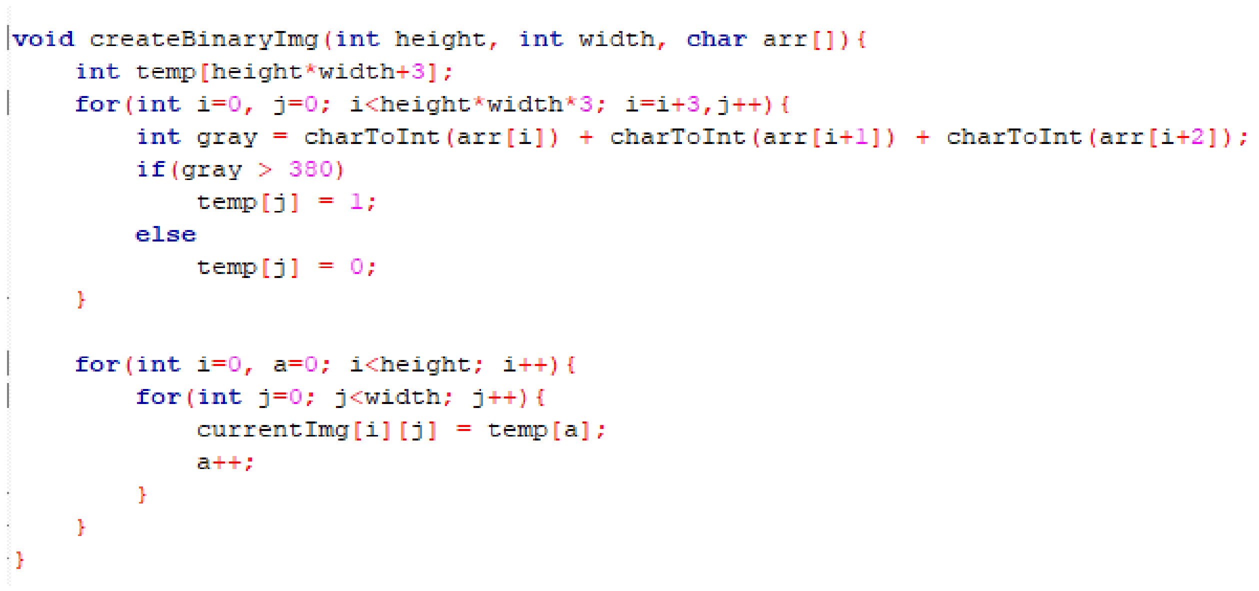


```
img = fopen(fName, "rb");

if(!img) {
    cout<< "Could not open file" <<endl;
    return ;
}

fseek(img, 0, SEEK_END);
int length = ftell(img);
fseek(img, 0, SEEK_SET);

fread(head, 1, 54, img);
int height = head[18];
int width = head[22];

char arr[height*width*3];

fread(arr, 1, height*width*3, img);
fclose(img);
```

Figure 5: Reading binary image

After I read the image, my work was to create a binary image of it.

In order to find the binary image I calculated the RGB values of each pixel and categorized them into 1 or 0 based a margin of RGB summation value.



```
void createBinaryImg(int height, int width, char arr[]){
    int temp[height*width+3];
    for(int i=0, j=0; i<height*width*3; i=i+3,j++){
        int gray = charToInt(arr[i]) + charToInt(arr[i+1]) + charToInt(arr[i+2]);
        if(gray > 380)
            temp[j] = 1;
        else
            temp[j] = 0;
    }

    for(int i=0, a=0; i<height; i++){
        for(int j=0; j<width; j++){
            currentImg[i][j] = temp[a];
            a++;
        }
    }
}
```

Figure 6: Creating Binary Image

As a result, out input to output transformation was like this -

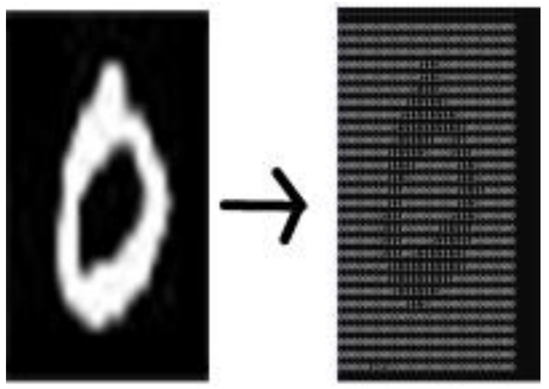
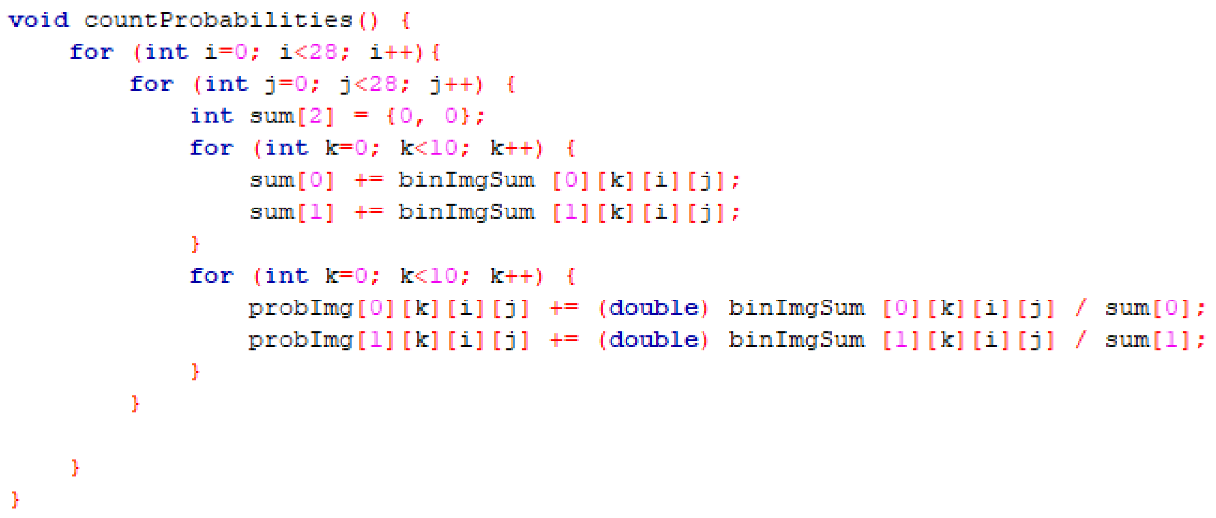


Figure 7: Binary image conversion

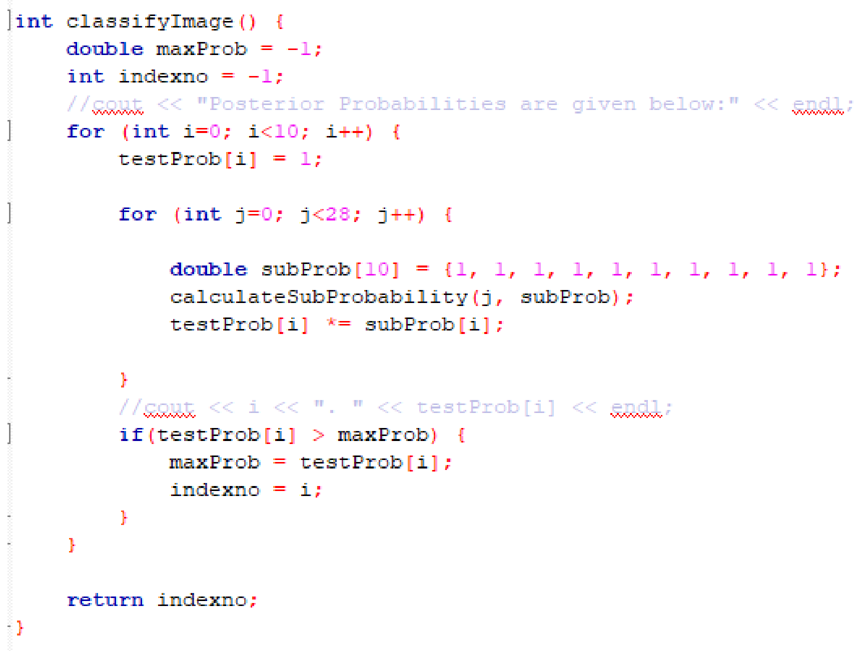
2.2 Digit Recognition using Naïve Bayes

In order to implement Naïve Bayer, at first I trained my system. I counted the sum of 1s and 0s of a digit and stored in an array. Then I calculated the probability of 0 or 1 in each pixel. Then we use the probabilities to calculate sub Probabilities of a single image. Afterwards, I classified the digit according to the sub Probability. Then, I chose the class, which had the most probability.



```
void countProbabilities() {
    for (int i=0; i<28; i++){
        for (int j=0; j<28; j++) {
            int sum[2] = {0, 0};
            for (int k=0; k<10; k++) {
                sum[0] += binImgSum [0][k][i][j];
                sum[1] += binImgSum [1][k][i][j];
            }
            for (int k=0; k<10; k++) {
                probImg[0][k][i][j] += (double) binImgSum [0][k][i][j] / sum[0];
                probImg[1][k][i][j] += (double) binImgSum [1][k][i][j] / sum[1];
            }
        }


    }
}
```

Figure 8: Count Probabilities



```
int classifyImage() {
    double maxProb = -1;
    int indexno = -1;
    //cout << "Posterior Probabilities are given below:" << endl;
    for (int i=0; i<10; i++) {
        testProb[i] = 1;

        for (int j=0; j<28; j++) {

            double subProb[10] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
            calculateSubProbability(j, subProb);
            testProb[i] *= subProb[i];

        }
        //cout << i << ". " << testProb[i] << endl;
        if(testProb[i] > maxProb) {
            maxProb = testProb[i];
            indexno = i;
        }
    }

    return indexno;
}
```

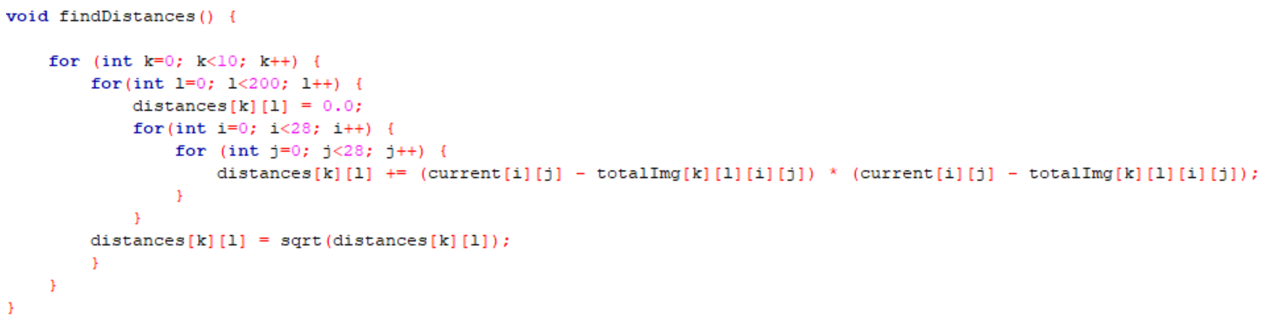
Figure 9: Determining digit class

2.3. Digit Recognition using K-Nearest-Neighbors:

K-nearest-neighbors measure the distance between two images. For distance measurement there are mainly 3 difference measurement methods:

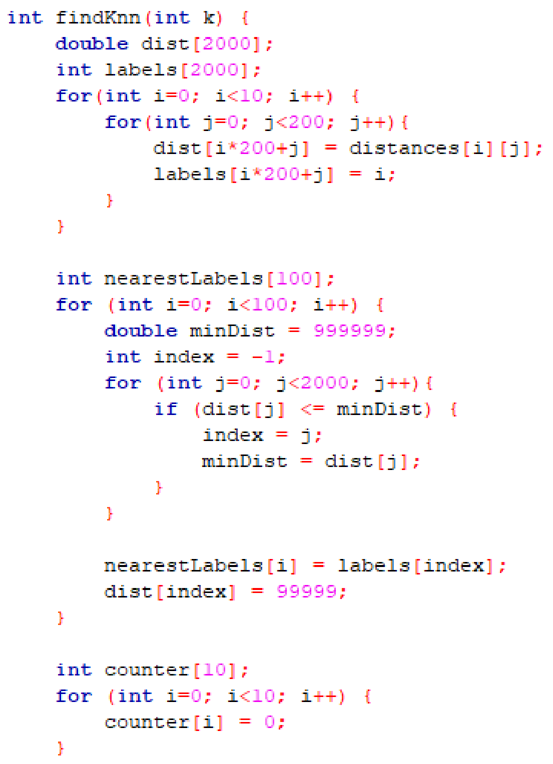
1. Euclidian 2. Manhattan 3. Minkowski

For my project I used Euclidian method. I stored my training image samples in an array. Afterwards, I calculated the pixel distances between the pixels of each of the sample images. Then, I took a value k, which is the minimum number of k distances. Then, my program voted for the most number of label/class found in the k distance data. The label with the highest vote is my estimated digit.



```
void findDistances() {

    for (int k=0; k<10; k++) {
        for(int l=0; l<200; l++) {
            distances[k][l] = 0.0;
            for(int i=0; i<28; i++) {
                for (int j=0; j<28; j++) {
                    distances[k][l] += (current[i][j] - totalImg[k][l][i][j]) * (current[i][j] - totalImg[k][l][i][j]);
                }
            }
        distances[k][l] = sqrt(distances[k][l]);
        }
    }
}
```

Figure 10: Find distance between images

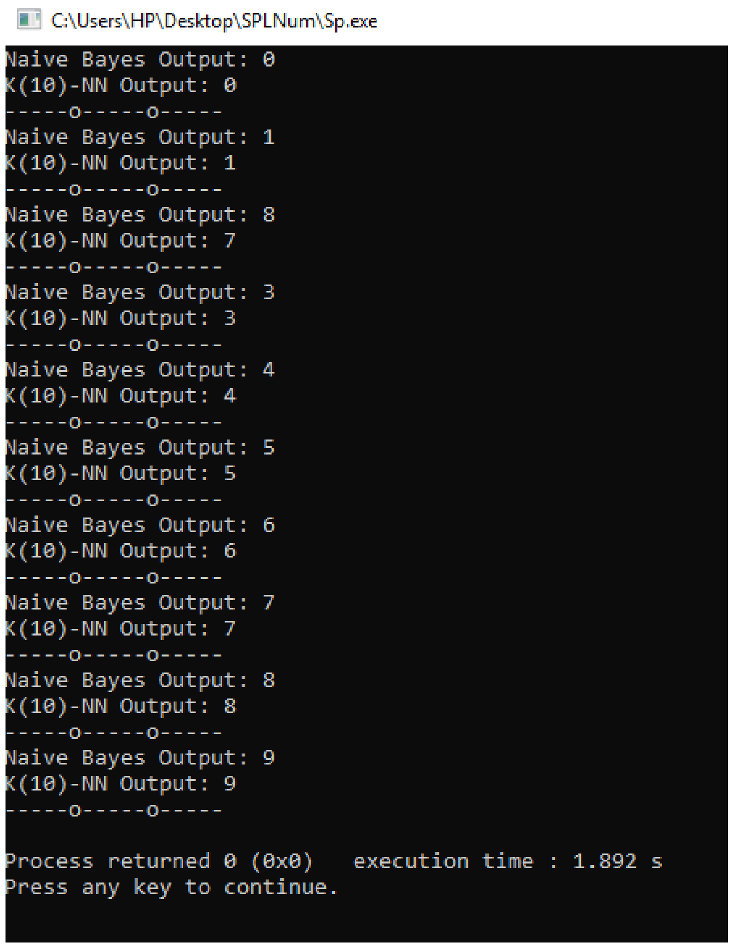


```
int findKnn(int k) {
    double dist[2000];
    int labels[2000];
    for(int i=0; i<10; i++) {
        for(int j=0; j<200; j++){
            dist[i*200+j] = distances[i][j];
            labels[i*200+j] = i;
        }
    }

    int nearestLabels[100];
    for (int i=0; i<100; i++) {
        double minDist = 999999;
        int index = -1;
        for (int j=0; j<2000; j++){
            if (dist[j] <= minDist) {
                index = j;
                minDist = dist[j];
            }
        }

        nearestLabels[i] = labels[index];
        dist[index] = 99999;
    }

    int counter[10];
    for (int i=0; i<10; i++) {
        counter[i] = 0;
    }
```

Figure 11: Find Nearest Neighbor

3. Output:

For this project, I have very limited contribution of the users. If we take inputs from 10 different digits {100021, 101021, 102021, 103021, 104021, 105021, 106021, 107021, 108021, 109021} the output will be the followings:



```
C:\Users\HP\Desktop\SPLNum\Sp.exe
Naive Bayes Output: 0
K(10)-NN Output: 0
-----o-----o-----
Naive Bayes Output: 1
K(10)-NN Output: 1
-----o-----o-----
Naive Bayes Output: 8
K(10)-NN Output: 7
-----o-----o-----
Naive Bayes Output: 3
K(10)-NN Output: 3
-----o-----o-----
Naive Bayes Output: 4
K(10)-NN Output: 4
-----o-----o-----
Naive Bayes Output: 5
K(10)-NN Output: 5
-----o-----o-----
Naive Bayes Output: 6
K(10)-NN Output: 6
-----o-----o-----
Naive Bayes Output: 7
K(10)-NN Output: 7
-----o-----o-----
Naive Bayes Output: 8
K(10)-NN Output: 8
-----o-----o-----
Naive Bayes Output: 9
K(10)-NN Output: 9
-----o-----o-----

Process returned 0 (0x0)   execution time : 1.892 s
Press any key to continue.
```

Figure 12: Output

4. Conclusion:

This whole project was very interesting. I faced many difficulties and found my way out. This Introduction to Machine Learning will help me for my further studies. It will be the building block of my machine learning understanding.

5. Reference

1. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
2. https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26
3. https://www.saedsayad.com/naive_bayesian.htm
4. http://www.dragonwins.com/domains/getteched/bmp/bmpfileformat.htm
5. https://www.geeksforgeeks.org/naive-bayes-classifiers/
6. https://en.wikipedia.org/wiki/Image_file_formats