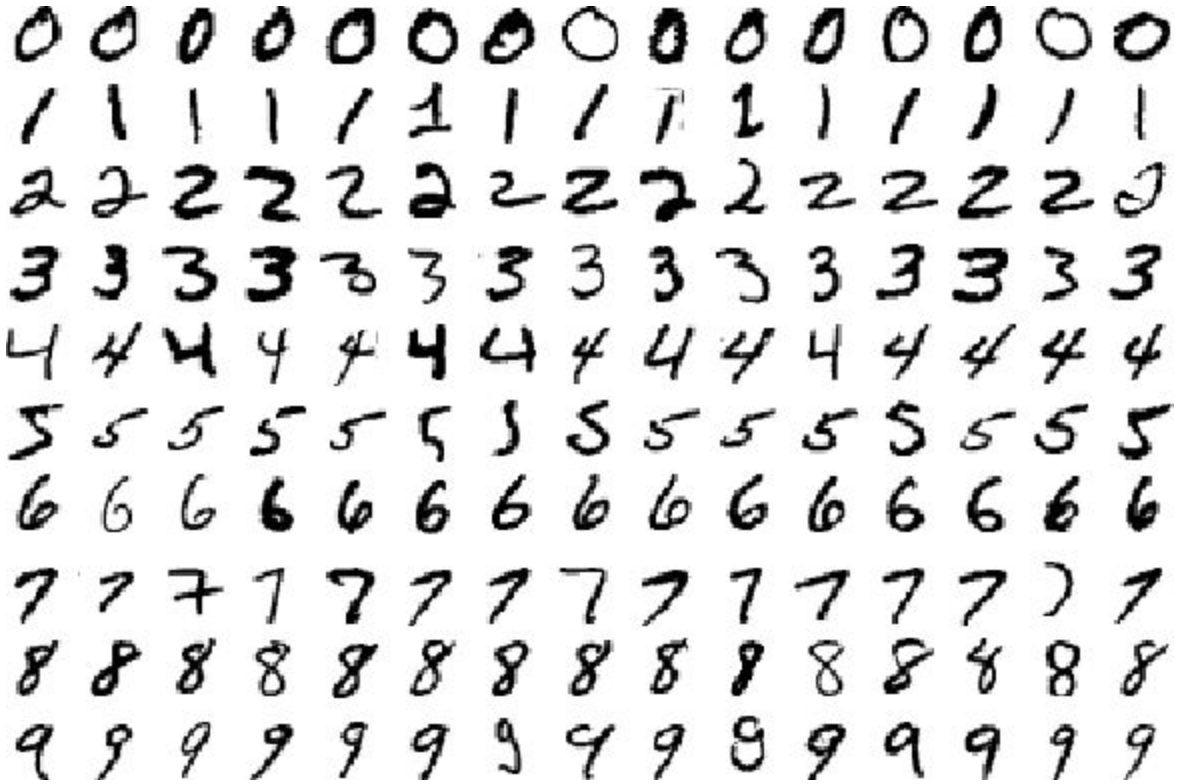
PROBLEM SET 1 REPORT

Introduction to Pattern Recognition- CSE-555



Jayant Solanki

UBIT Name: jayantso
Person Number: 50246821
Fall 2017 CSE

INTRODUCTION

This report contains the implementation details of Problem Set 1 i.e., classifying handwritten digit image using 0/1 Loss function with Bayesian Decision. We will train our model using the training data sets ("train-images-idx3-ubyte.gz" and "train-labels-idx1-ubyte.gz") and test the performance using the test data set ("t10k-images-idx3-ubyte.gz" and "t10k-labels-idx1-ubyte.gz"). MNIST data has been taken from (http://yann.lecun.com/exdb/mnist).

OBJECTIVES ACHIEVED

We have trained our Gaussian Naive Bayesian model on the MNIST Training set and predicted the labels of the digit images in MNIST Test dataset.

- Mean and Standard Deviation of MNIST Train images:
    - For each digit category we calculated the mean and standard deviation values of each features and plotted the image in 28x28 pixels and called them called them "**mean digits**" and "**standard deviation digits**".
- Gaussian Naive Bayesian Classifier:
    - Assuming that each feature is independent with one another and have Gaussian distribution, we trained the model on training set and achieved an accuracy of **56.49%** on MNIST train images and **55.58%** on **MNIST test images**.
    - We can see that the overall accuracy is way lower than the accuracy provided by other classifiers such as CNN, Single Layer Neural Network or Logistic Regression, the reason is because Gaussian Naive Bayesian classifier is known to be a bad estimator, and assumes each feature to be independent to one another which is not always in case for MNIST dataset.

IMPLEMENTATION

BACKGROUND[1]

Naive Bayes methods are a set of supervised learning algorithms based on applying

[1] H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector $x_1$through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive independence assumption that

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and

spam filtering. **They require a small amount of training data to estimate the necessary parameters**. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

On the flip side, although naive Bayes is known as a decent classifier, **it is known to be a bad estimator**, so the probability outputs are not to be taken too seriously.

Assuming the feature vectors in each category in the training data "train-images-idx3-ubyte.gz") have Gaussian distribution we have used Gaussian Naive Bayes Model approach using **Sklearn Python Library**.

1. **Data Set Reading:**
    a. MNIST dataset files were downloaded from the website mentioned in the instructions and have been read using the **gzip** library function of the python with the help of direction mentioned in this website.
    b. Images' data have been stored in 60000x28x28 size in Trains_images numpy array and 10000x28x28 size in Test_images numpy array.
    c. Images' label have been stored in 60000x1 size in trains_images_label numpy array and 10000x1 test_images_label.
2. **Drawing of mean digits and standard deviation digits:**
    a. We used where function of numpy library to find out indices of images having a particular digit class. We then iterate it through all the digit classes and calculated the mean and standard deviation of features in row wise fashion.
    b. The resultant 28x28 images for both mean and standard deviation for each class were saved as jpeg images.
    c. Those images are stored under **mean-digits** and s**tandard-deviation-digits** folder under code directory.
3. **Gaussian Naive Bayesian Classifier:**
    a. Train and Test Images were then flattened into 2D numpy array, N x 784

size.

b. Data has not been standardised because, standardisation leads to lower accuracy in case for the current implementation of Gaussian Naive Bayes classifier.
c. Using the **GaussianNB** function of **Sklearn library** we created the model for Gaussian Naive Bayes Classifier.
d. The model was then trained using the training set and was then checked for performance on the test data set.

DIRECTORY LAYOUT

There are 3 folders under **Assignment-1** directory:

- **code :**
    1. **MNIST_Data folder:** Contains the gz files of Train and Test images data sets along with their labels.
    2. **main.py**: Run this python3 file for execution of Part 1 and Part 2 of the Problem Set 1.
    3. **Part-1-output folder**: Contains 20 images of mean-digits and standard-deviation-digits for all 10 digit classes.
    4. **libs.py**:
        a. **read_gz(images,labels)**: To read MNIST gz data
        b. **view_image(image, label="")**: to view single image from the MNIST data and save the concerned mean-digits and standard-deviation-digits jpeg images.
- **Outputs:** Contains 20 images of mean-digits and standard-deviation-digits for all 10 digit classes.
- **Report:** Contains the word doc file of the documentation for Problem Set 1.

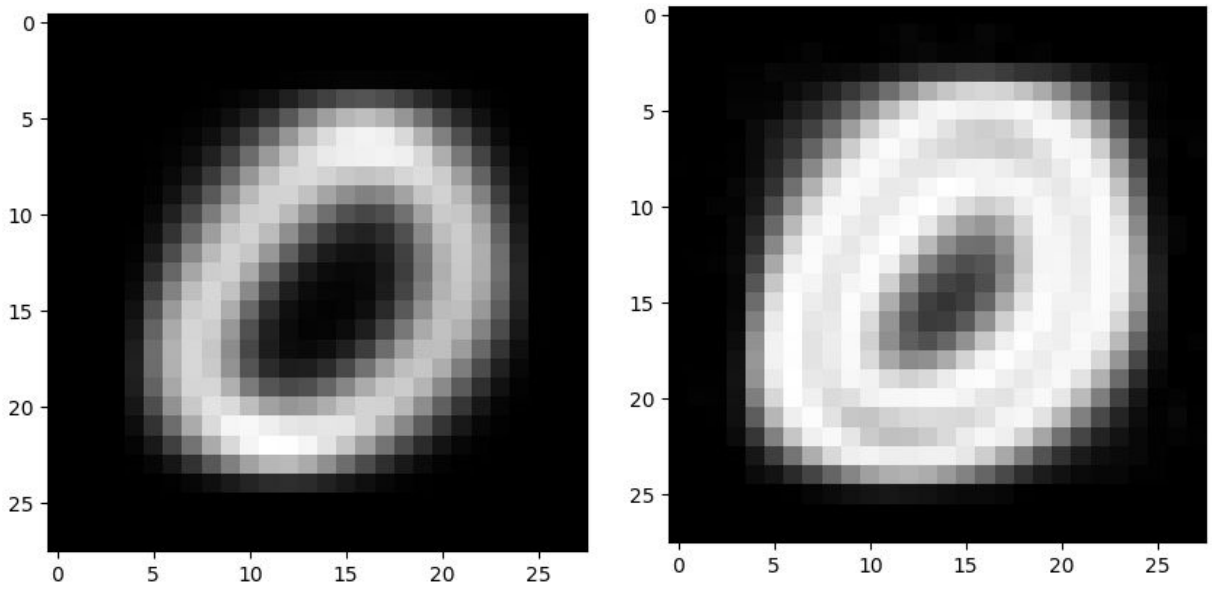
IMPROVEMENTS

MNIST Data Set: The training and test dataset after reading from the gzip file were stored in the file systems as numpy files and were later reload every time whenever needed. This saved us from unnecessary load time and processing time.

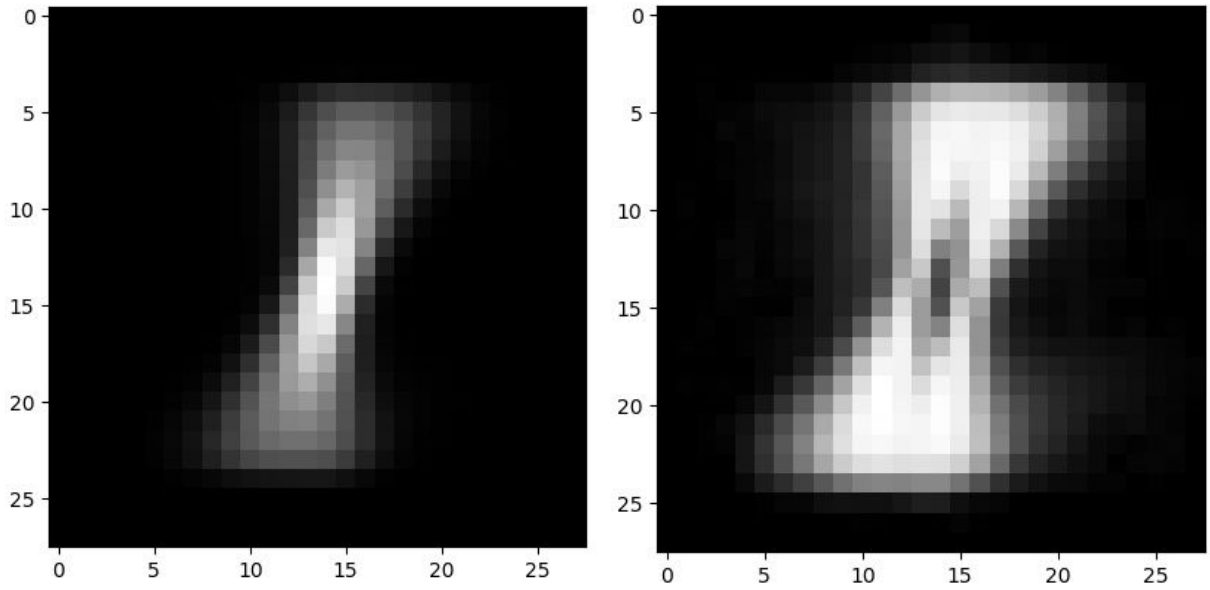
RESULTS

1. **Part 1: 20 images**

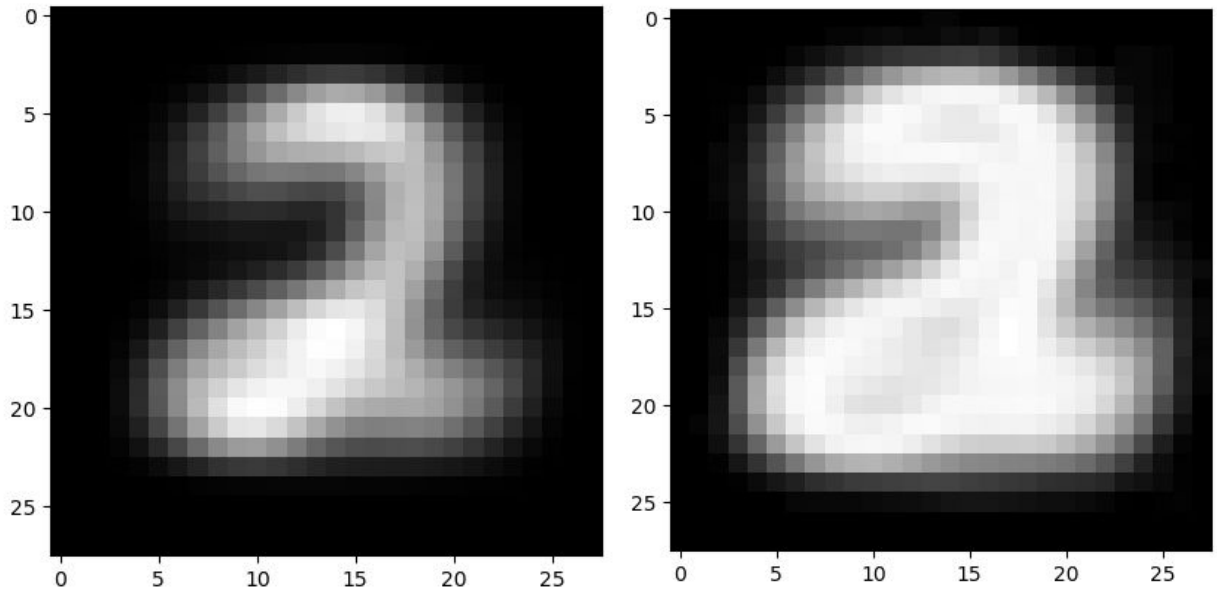
Mean-digits **Standard-deviation-digits**



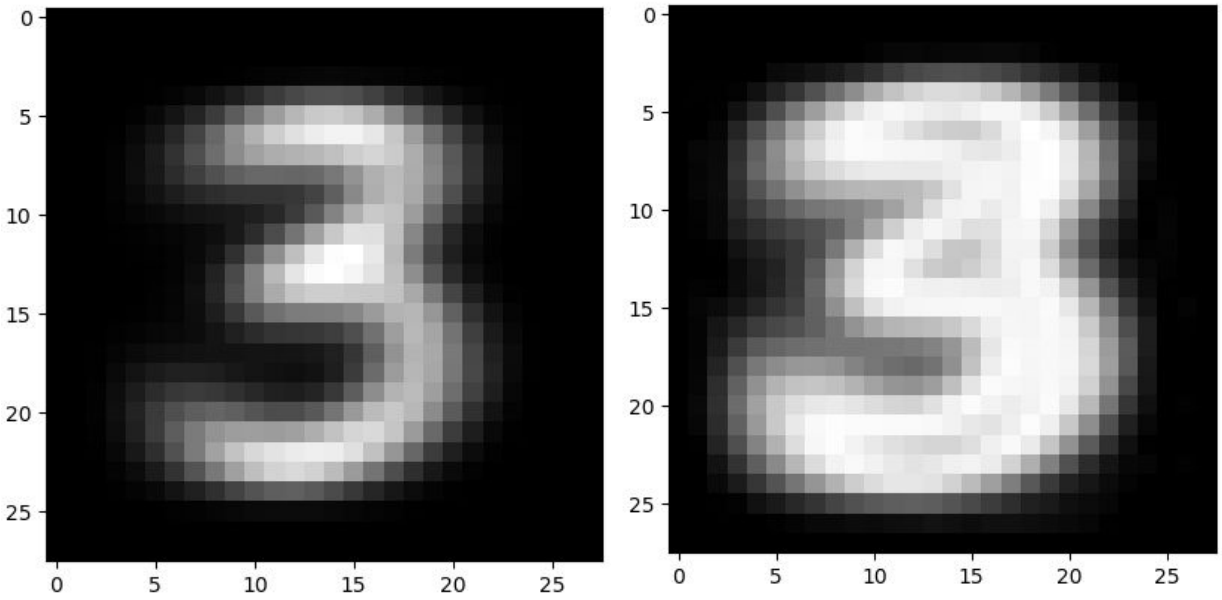
Zero



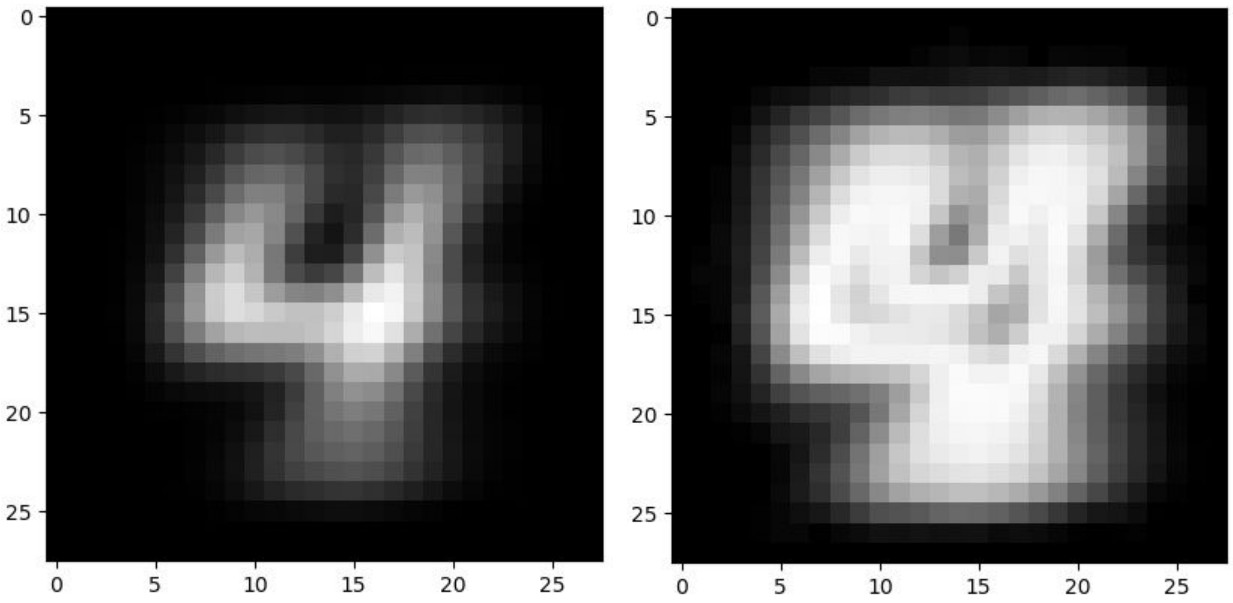
One


0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25

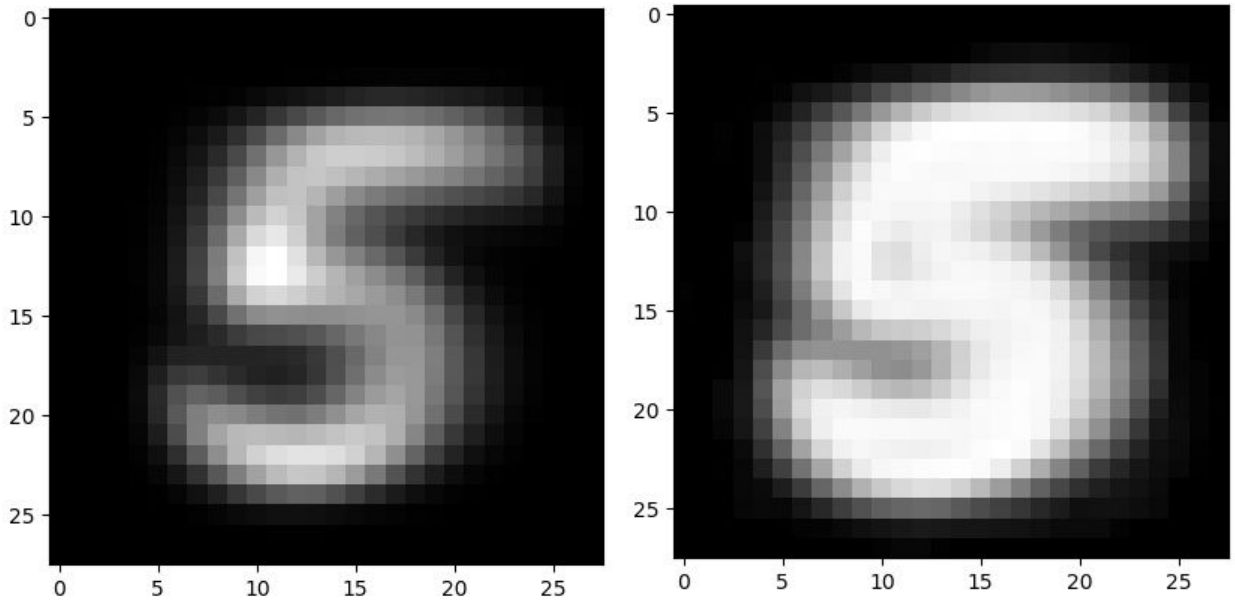
Two


0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25

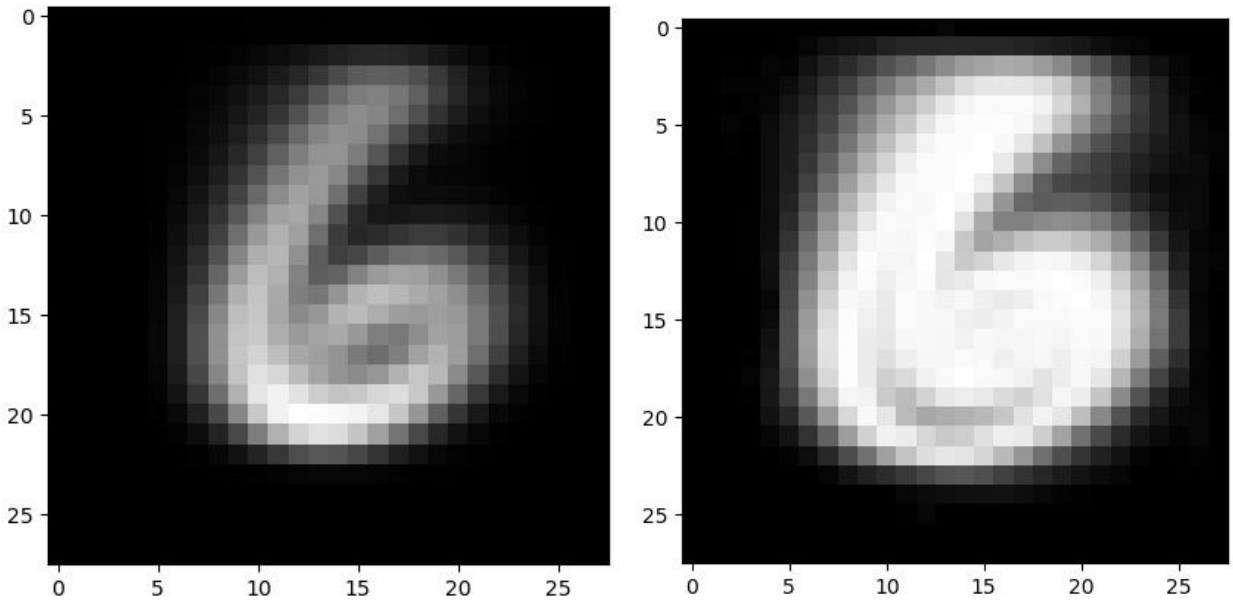
Three


0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25

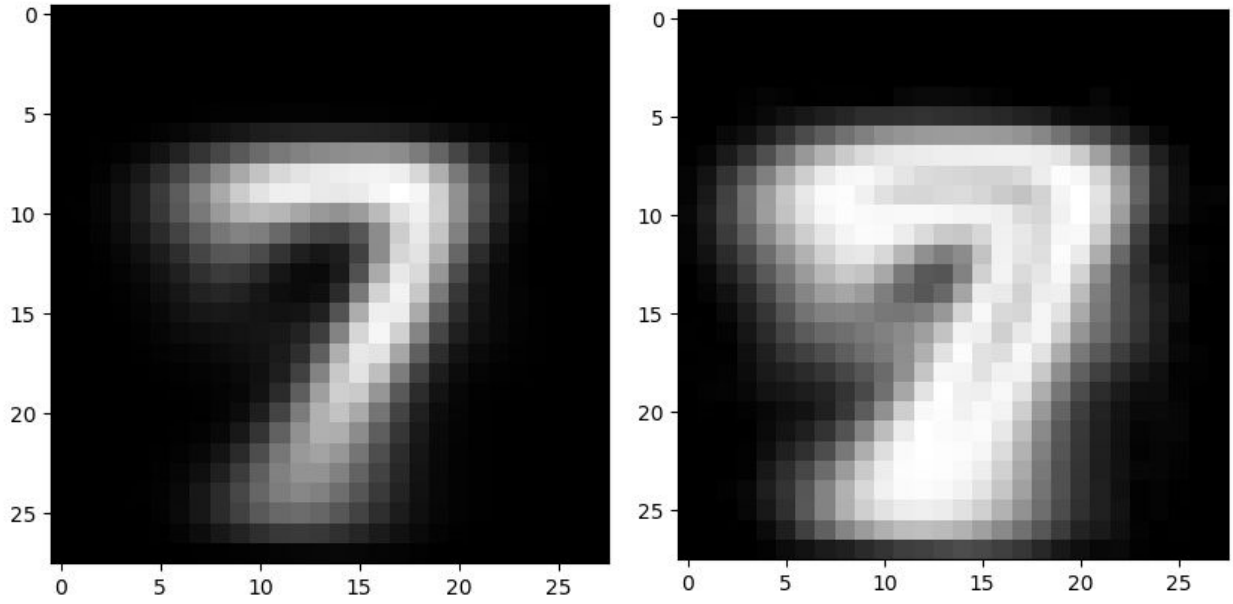
Four


0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25
0
5
10
15
20
25

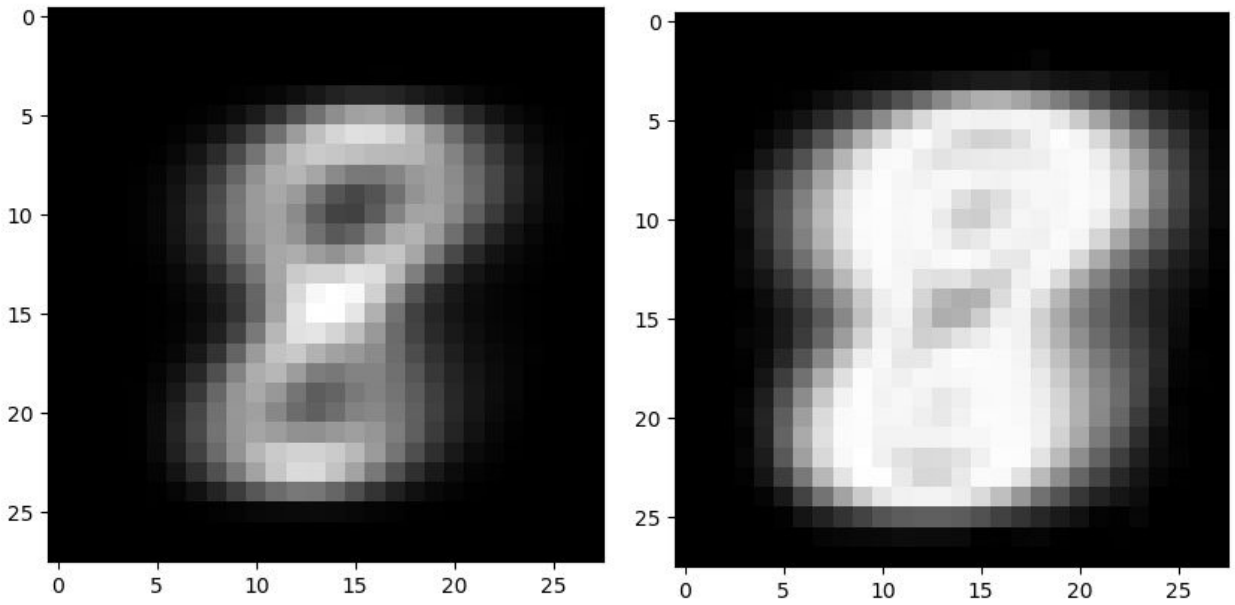
Five



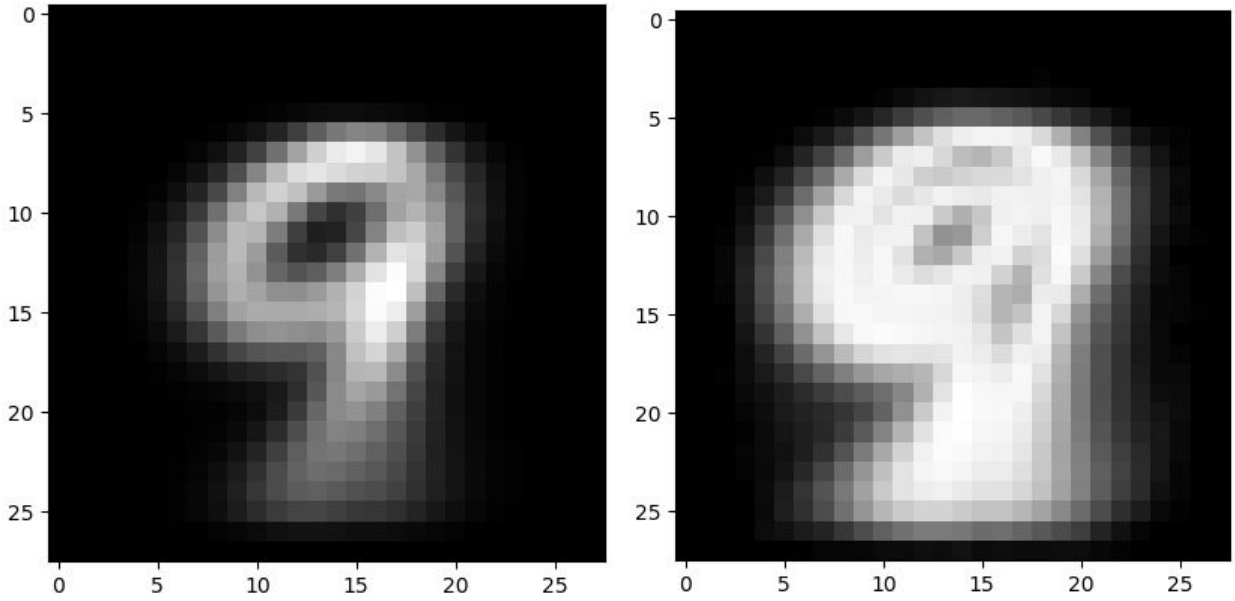
Six



Seven



Eight



Nine

2. **Part 2: 0/1 Loss Gaussian Naive Bayesian Classifier**

Datasheet: https://goo.gl/etb46A

Train Data Result				
Digits	**precision**	**recall**	**f1-score**	**support**
0	0.77	0.88	0.82	5923
1	0.81	0.94	0.87	6742
2	0.88	0.3	0.45	5958
3	0.74	0.37	0.49	6131
4	0.84	0.18	0.29	5842
5	0.66	0.06	0.11	5421
6	0.67	0.94	0.78	5918
7	0.91	0.31	0.46	6265
8	0.29	0.65	0.4	5851
9	0.38	0.95	0.54	5949
avg / total	**0.7**	**0.56**	**0.53**	**60000**
Test Data Results				
Digits	**precision**	**recall**	**f1-score**	**support**
0	0.79	0.89	0.84	980
1	0.85	0.95	0.9	1135
2	0.9	0.26	0.4	1032
3	0.71	0.35	0.47	1010
4	0.88	0.17	0.29	982
5	0.55	0.05	0.09	892
6	0.65	0.93	0.77	958
7	0.88	0.27	0.42	1028
8	0.28	0.67	0.4	974
9	0.37	0.95	0.53	1009
avg / total	**0.69**	**0.56**	**0.52**	**10000**

Training set Accuracy is 0.564900
Test set Accuracy is 0.555800

SOFTWARE/HARDWARE USED

- Sublime Text 3,
- Python 3 Environment based on Anaconda,
- Ubuntu 17 System, Intel core i7 processor

- Python libraries: numpy, matplotlib, sklearn and pylab

REFERENCES

1. Ublearns
2. Stackoverflow.com
3. Python, Numpy and Sklearn documentations
4. http://yann.lecun.com/exdb/mnist/
5. https://martin-thoma.com/classify-mnist-with-pybrain/
6. http://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes
7. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB
8. H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.