In The name of God

University Tehran

Engineering Facility

Electrical and Computer Engineering

Pattern Recognition

HW # 3

Amin Fadaeinedjad

810195442

Spring 98

List

Abstract

This Homework is about using different methods to learn a dataset and predict the test dataset in and we also compared it with the ready code in the scikit-learn library and we saw the result of the two classification and we also plotted the confusion matrix and the confidence matrix of this result. We also want to compare the classifiers with each other and we also going to see the tradeoff between accuracy and time.

Question 1

In this part we only have two result as yes or no and we need to calculate the probability of the cases and by looking at the data set we can see that in the 10 occasions that have been sawed 5 cases was a yes and 5 cases were no and that makes the $P(v_i) = P(v_2) = 0.5$ and we also know that $P\left(a_i \middle| v_j\right) = \frac{n_c+mp}{n+m}$ and by looking at the data we can calculate the probability we also define a_i vector as a 3x1 dimension vector which is $[color\ Type\ Origin]$ and the probability and we also know that the vector has independent data so

$$P\left([color\ Type\ Origin]\middle|v_j\right) = P\left(color\middle|v_j\right)P\left(Type\middle|v_j\right)P(Origin|v_j)$$

This are the results $v_1 = yes\ \ v_2 = no$ which m is the number of samples that is 10 and p is the prior knowledge that we have from the data

$$P(red|yes) = \frac{3 + 10 * \left(\frac{3}{5}\right)}{5 + 10} = \frac{3}{5} \quad P(red|no) = \frac{2 + 10 * \left(\frac{2}{5}\right)}{5 + 10} = \frac{2}{5}$$

$$P(yellow|yes) = \frac{2 + 10 * \left(\frac{2}{5}\right)}{5 + 10} = \frac{2}{5} \quad P(yellow|no) = \frac{3 + 10 * \left(\frac{3}{5}\right)}{5 + 10} = \frac{3}{5}$$

$$P(sports|yes) = \frac{4 + 10 * \left(\frac{4}{5}\right)}{5 + 10} = \frac{4}{5} \quad P(sport|no) = \frac{2 + 10 * \left(\frac{2}{5}\right)}{5 + 10} = \frac{2}{5}$$

$$P(suv|yes) = \frac{\left(1 + 10 * \left(\frac{1}{5}\right)\right)}{5 + 10} = \frac{1}{5} \quad P(suv|no) = \frac{3 + 10 * \left(\frac{3}{5}\right)}{5 + 10} = \frac{3}{5}$$

$$P(Domestic|yes) = \frac{2 + 10 * \left(\frac{2}{5}\right)}{5 + 10} = \frac{2}{5} \quad P(Domestic|no) = \frac{3 + 10 * \left(\frac{3}{5}\right)}{5 + 10} = \frac{3}{5}$$

$$P(imported|yes) = \frac{3 + 10 * \left(\frac{3}{5}\right)}{5 + 10} = \frac{3}{5} \quad P(imorted|no) = \frac{2 + 10 * \left(\frac{2}{5}\right)}{5 + 10} = \frac{2}{5}$$

So now we have to calculate the probability of the Red Domestic SUV probability that whether it is stolen or not so:

$$V_{nb} = argmax_{v_j=\{yes,no\}}P\left(v_j\right)P\left([color, Type, Origin]\middle|v_j\right)$$

We only have two v_j so we only need to compare this two

We know that $v_{yes} = v_{no} = 0.5$

$$P\left([color\ Type\ Origin]\middle|v_j\right) = P\left(color\middle|v_j\right)P\left(Type\middle|v_j\right)P(Origin|v_j)$$

$$P(Red\ SUV\ domestic|yes) = P(red|yes)P(suv|yes)P(domestic|yes)$$

$$P(red\ SUV\ domestic|no) = P(red|no)P(SUV|no)P(domestic|no)$$

$$P(red\ suv\ domestic|yes) = \frac{3}{5} \times \frac{1}{5} \times \frac{2}{5} = \frac{6}{125}$$

$$P(red\ suv\ domestic|no) = \frac{2}{5} \times \frac{3}{5} \times \frac{3}{5} = \frac{18}{125}$$

So $P(red\ suv\ domestic|no) > P(red\ suv\ domestic|yes)$

So it is more likely to be stolen

And we showed in this case without having any date from a case we can guess whether it is stolen or not.

Question 2

$$p_n(x) = \frac{1}{nh_n}\sum_{i=1}^{n} \varphi(\frac{x - x_i}{h_n}) \quad p(x) = N(\mu, \sigma)$$

1)

$$E\{p_n(x)\} = \frac{1}{nh_n}\sum_{i=1}^{n} E\left\{\varphi\left(\frac{x - x_i}{h_n}\right)\right\} \to E\left\{\varphi\left(\frac{x - x_i}{h_n}\right)\right\} = \int \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h_n^2}} dx_i$$

$$\int p(x|x_i)p(x_i|D)dx_i = \int \frac{1}{2\pi\sigma} \exp\left[-\frac{1}{2}\left(\frac{x_i - \mu}{\sigma}\right)^2 - \frac{1}{2}\left(\frac{x - x_i}{h_n}\right)^2\right]$$

$$\exp(-\frac{1}{2}\left(\frac{x_i^2 - 2x_i\mu + \mu^2}{\sigma^2} + \frac{x^2 - 2x\, x_i + x_i^2}{h_n^2}\right) \to$$

$$\exp\left(-\frac{1}{2}\left(\frac{x^2}{h_n^2} + \frac{\mu^2}{\sigma^2}\right) + \frac{\left(\frac{x}{h_n^2} + \frac{\mu}{\sigma^2}\right)^2}{\frac{2(h_n^2+\sigma^2)}{h_n^2\sigma^2}}\right) \exp\left(-\frac{1}{2}\left(\frac{\frac{\left(x_i - \frac{h_n^2\sigma^2}{h_n^2+\sigma^2}\left(\frac{x}{h_n^2} + \frac{\mu}{\sigma^2}\right)\right)}{2(h_n^2\sigma^2)}}{h_n^2 + \sigma^2}\right)^2\right)$$

$$\to \frac{h_n\sigma}{\sqrt{h_n^2 + \sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x^2}{h_n^2} + \frac{\mu^2}{\sigma^2} - \frac{h_n^2\sigma^2\left(\frac{x}{h_n^2} + \frac{\mu}{\sigma^2}\right)^2}{h_n^2 + \mu^2}\right)\right) \to$$

$$\exp\left(-\frac{1}{2}\left(\frac{x^2\sigma^2 + \mu^2 h_n^2}{h_n^2\sigma^2} - \frac{\left(h_n^2\sigma^2\left(\frac{x}{h_n^2} + \frac{\mu}{\sigma^2}\right)^2\right)}{h_n^2 + \sigma^2}\right)\right) \to$$

$$\exp\left(-\frac{1}{2}\left(\frac{x^2\sigma^4 + \mu^2 h_n^4 + x^2 h_n^2\sigma^2 + \mu^2 h_n^2\sigma^2}{h_n^2\sigma^2(h_n^2 + \sigma^2)} - \frac{h_n^4\sigma^4\left(\frac{x^2}{h_n^4} + 2\frac{x\mu}{h_n^2\sigma^2} + \frac{\mu^2}{\sigma^4}\right)^1}{h_n^2\sigma^2(h_n^2 + \sigma^2)}\right)\right) \to$$

$$\exp(-\frac{1}{2}\left(\frac{(x^2 h_n^2\sigma^2 + \mu^2 h_n^2\sigma^2)}{h_n^2\sigma^2(h_n^2 + \sigma^2)} - \frac{2x\mu}{h_n^2 + \sigma^2}\right) = \exp\left(-\frac{1}{2}\left(\frac{(x - \mu)^2}{h_n^2 + \sigma^2}\right)\right) \to$$

$$= \frac{1}{\sqrt{2\pi(h_n^2+\sigma^2)}} \exp\left[-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2+h_n^2}\right)\right] \to \overline{p(x)} \sim N(\mu, \sigma^2+h_n^2)$$

2) $Var[p_n(x)] = Var\left[\frac{1}{n^2h_n^2}\sum_{i=1}^{n} \varphi\left(\frac{x-x_i}{h_n}\right)\right]$ we can take out the sigma a put a vector

$$Var\left[\frac{1}{nh_n^2}\varphi\left(\frac{x-v}{h_n}\right)\right] = \frac{1}{nh_n^2}Var\left[\varphi\left(\frac{x-v}{h_n}\right)\right]$$

$$= \left[E\left\{\frac{1}{nh_n^2}\varphi^2\left(\frac{x-v}{h_n}\right)\right\} - \left(E\left\{\frac{1}{nh_n^2}\varphi\left(\frac{x-v}{h_n}\right)\right\}\right)\right]$$

$$\varphi(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right) \to$$

$$E\left\{\frac{1}{nh_n^2}\varphi^2\left(\frac{x-v}{h_n}\right)\right\}$$

$$= \frac{1}{nh_n^2}\int_{-\infty}^{\infty}\frac{1}{2\pi}\exp\left(-\left(\frac{x-v}{h_n}\right)^2\right)\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{(v-\mu)^2}{\sigma^2}\right)\right)dv \to$$

We can use the past part

$$\frac{1}{\sqrt{2\pi}nh_n^2}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x-v}{\frac{h_n}{\sqrt{2}}}\right)^2\right)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{(v-\mu)^2}{\sigma^2}\right)\right)dv \to$$

$$\frac{1}{\sqrt{2\pi}nh_n^2}N\left(\mu, \frac{h_n^2}{2}+\sigma^2\right) = \frac{1}{nh_n^2}\frac{1}{2\pi\sqrt{2\left(\frac{h_n^2}{2}+\sigma^2\right)}}\exp\left(-\frac{1}{2}\left(\frac{(x-\mu)^2}{\frac{h_n^2}{2}+\sigma^2}\right)\right)$$

$$E\left\{\varphi\left(\frac{(x-v)^2}{\sigma^2}\right)\right\} = nh_n\frac{1}{\sqrt{2\pi(h_n^2+\sigma^2)}}\exp(-\frac{1}{2}\left(\frac{(x-\mu)^2}{h_n^2+\sigma^2}\right) \to$$

$$\frac{1}{n^2h_n^2}E\left\{\varphi\left(\frac{(x-v)^2}{\sigma^2}\right)\right\} = \frac{1}{2\pi(h_n^2+\sigma^2)}\exp\left(-\frac{(x-\mu)^2}{h_n^2+\sigma^2}\right)$$

$$Var[p_n(x)] \simeq \frac{1}{2nh_n\sqrt{\pi}}\,\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\frac{1}{2}(x-v)^2}{\sigma^2}\right)$$

$$Var[p_n(x)] \simeq \frac{1}{2nh_n\sqrt{\pi}}p(x)$$

3) $p(x) - \overline{p_n(x)} = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2}\right)\right) - \frac{1}{\sqrt{2\pi(\sigma^2+h_n^2)}}\exp\left(-\frac{1}{2}\left(\frac{(x-\mu)^2}{\sigma^2+h_n^2}\right)\right)$

$$= p(x)\left[1 - \frac{\sigma}{\sqrt{\sigma^2+h_n^2}}\exp\left(-\frac{(x-\mu)^2}{2}\left(\frac{1}{\sigma^2+h_n^2}-\frac{1}{\sigma^2}\right)\right)\right] =$$

$$p(x)\left[1-\frac{\sigma}{\sigma\left(1+\frac{h_n^2}{\sigma^2}\right)^{\frac{1}{2}}}\exp\left(-\frac{(x-\mu)^2}{2}\left(-\frac{h_n^2}{\sigma^2(\sigma^2+h_n^2)}\right)\right)\right]=$$

$$p(x)\left[1-\left(1-\frac{1}{2}\left(\frac{h_n}{\sigma}\right)^2\right)\left(1+\frac{(x-\mu)^2h_n^2}{2\sigma^2(\sigma^2+h_n^2)}\right)\right]\rightarrow$$

We now will only show the parts that have h_n^2 in them

$$p(x)\left[1-\left(1-\frac{1}{2}\left(\frac{h_n}{\sigma}\right)^2+\frac{(x-\mu)^2}{2}\frac{h_n^2}{\sigma^2(\sigma^2+h_n^2)}-\frac{(x-\mu)^2h_n^4}{4\sigma^2(\sigma^4+\sigma^2h_n^2)}\right)\right]\rightarrow$$

We now know that $h_n \ll \sigma \rightarrow \sigma^2+h_n^2 \simeq \sigma^2 \quad and \, \frac{h_n^4}{\sigma^4+\sigma^2h_n^2} \simeq 0$

$$p(x)\left[\frac{1}{2}\left(\frac{h_n}{\sigma}\right)^2\right]\left[1-\frac{(x-\mu)^2}{\sigma^2}\right]$$

So we have

$$p(x)-\overline{p_n(x)} \simeq p(x)\left(\frac{1}{2}\left(\frac{h_n}{\sigma}\right)^2\right)\left(1-\frac{(x-\mu)^2}{\sigma^2}\right)$$

Question 3

I) We need to calculate the parameters of the distributions that we have and the algorithm is something like this

$$\hat{\theta}=argMax\, f(\theta|x)\rightarrow f(\theta|x)f(x)=f(x|\theta)f(\theta)\rightarrow f(\theta|x)=\frac{f(x|\theta)f(\theta)}{f(x)}\rightarrow$$

$$f(x)\rightarrow constant\rightarrow\ \hat{\theta}=argMax\, f(x|\theta)f(\theta)\rightarrow\hat{\theta}\rightarrow\hat{\mu}\,\hat{\Sigma}$$

$$\hat{\theta}=argMax\,\prod f(x_i|\theta)f(\theta)\rightarrow$$

$$\prod\frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\right)f(\theta)\rightarrow$$

For easier calculation we are going to use logarithm

$$\sum(-\frac{n}{2}\log(2\pi)-\frac{1}{2}\log(|\Sigma|)-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)+\log\big(f(\theta)\big))=\mathcal{L}(\mu,\Sigma)\rightarrow$$

$$\frac{d\mathcal{L}(\mu,\Sigma)}{d\mu}=0\rightarrow\sum-\Sigma^{-1}(x_i-\mu)=0\rightarrow\sum x_i=Q\mu\rightarrow\mu=\frac{\sum x_i}{Q}$$

$$\frac{d\mathcal{L}(\mu,\Sigma)}{d\Sigma}=0\rightarrow\sum-\frac{1}{2\Sigma}+\frac{1}{2}(x_i-\mu)^T\Sigma^{-2}(x_i-\mu)=0\rightarrow$$

$$\frac{Q}{\Sigma}=\sum(x_i-\mu)^T\Sigma^{-2}(x_i-\mu)\rightarrow\times\Sigma^2\rightarrow Q\Sigma=\sum_{i=1}^{Q}\Sigma^2(x_i-\mu)^T\Sigma^{-2}(x_i-\mu)\rightarrow$$

The Σ matrix is a diagonal matrix so we can see that the Σ

$$\Sigma = \frac{1}{Q}\sum_{i=1}^{Q}(x_i - \mu)^T(x_i - \mu)$$

And the we should also determine the $f(\theta)$ of the distributions and we need a prior knowledge about this.

$f(\theta_i) = \frac{N_i}{N}$ which the N_i is the number samples that belong to class i and N is the number of all the samples and if we don't have any prior knowledge about the data we can assign that $f(\theta_i) = \frac{1}{C}$ which C is the number of classes that we have.

II) There are different reasons that we may face problem for our classification one of them is small sample size, in many pattern recognition applications there is a large number of features n and the number of training pattern N_i per class may be significantly less than the dimension of the feature space. This means that the covariance matrix will be singular and can't be inverted. Using PCA we can reduce these to a small number of principle components scores.

III) We can use some methods to apply to the classifier and getting out of the situation. One of this methods is Diagonal covariance estimators (van Ness) if we take a small sample size covariance estimate and set all the non-diagonal elements to zero then the resulting matrix will generally full rank. The van Neww estimator, intended for the non-parametric classifier, therefor simply set all the off diagonal elements to zero.

$$\sum_{i}^{vn}(\alpha) = \alpha \times diag(\Sigma_i)$$

The second method for debugging this situation is Regularization or shrinkage, the general idea of shrinkage is to stabilize a poor matrix estimate by bending it with a stable known matrix for example given s singular small sample size covariance matrix Σ_i we can make a full rank matrix by using this method.

$$\sum_{i}^{id}(\alpha) = (1 - \alpha)\Sigma_i + \alpha\sigma I$$

Where α is the shrinkage parameter and the σ is the average of the diagonal values

$$\sigma = \frac{1}{N}\sum trace(\Sigma_i)$$

Friedman proposed a composite shrinkage method called regularized discriminant analysis (RDA) that blends the class covariance estimate with both the pooled and the identity matrix. Shrinkage towards the pooled matrix is done by one scalar parameter λ .

$$\sum_{i}^{pool}(\lambda) = (1 - \lambda)\Sigma_i + \lambda\Sigma_p$$

Shrinkage towards the identity matrix is then calculated using a second scalar parameter γ.

$$\sum_{i}^{RDA}(\lambda,\gamma) = (1-\gamma)\Sigma_{\mathrm{i}}^{pool}(\lambda) + \gamma\sigma I$$

We used the regularization or shrinkage method for this question.

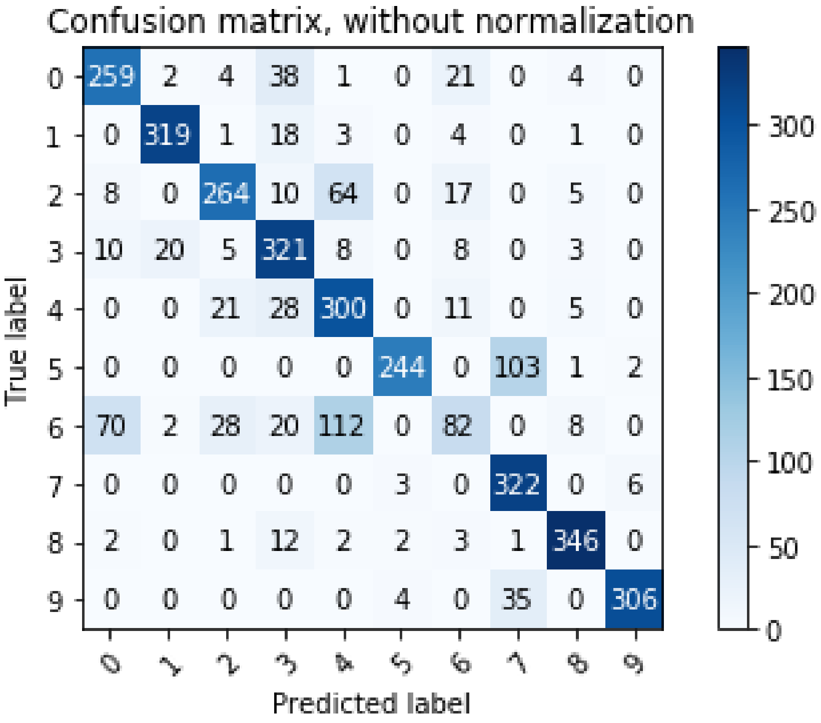
The result for the following method is mentioned below

At first we need to normalize the dataset that we have, and we have done it by multiplying it in 0.01 this makes the range of our data's reasonable

Time of train: 1.640625

Time of test: 126.515625

Accuracy: 78.94 %



	0	1	2	3	4	5	6	7	8	9
0	259	2	4	38	1	0	21	0	4	0
1	0	319	1	18	3	0	4	0	1	0
2	8	0	264	10	64	0	17	0	5	0
3	10	20	5	321	8	0	8	0	3	0
4	0	0	21	28	300	0	11	0	5	0
5	0	0	0	0	0	244	0	103	1	2
6	70	2	28	20	112	0	82	0	8	0
7	0	0	0	0	0	3	0	322	0	6
8	2	0	1	12	2	2	3	1	346	0
9	0	0	0	0	0	4	0	35	0	306

Figure 1

Figure 1 shows the confusion matrix of the pdf estimate with $\alpha = 0.019$ we need to define an α because in the previous part we saw that the determine of the covariance matrix become incredibly small and when we are using the inverse of that matrix then the result will becomes big and we will get wrong values from our calculation.

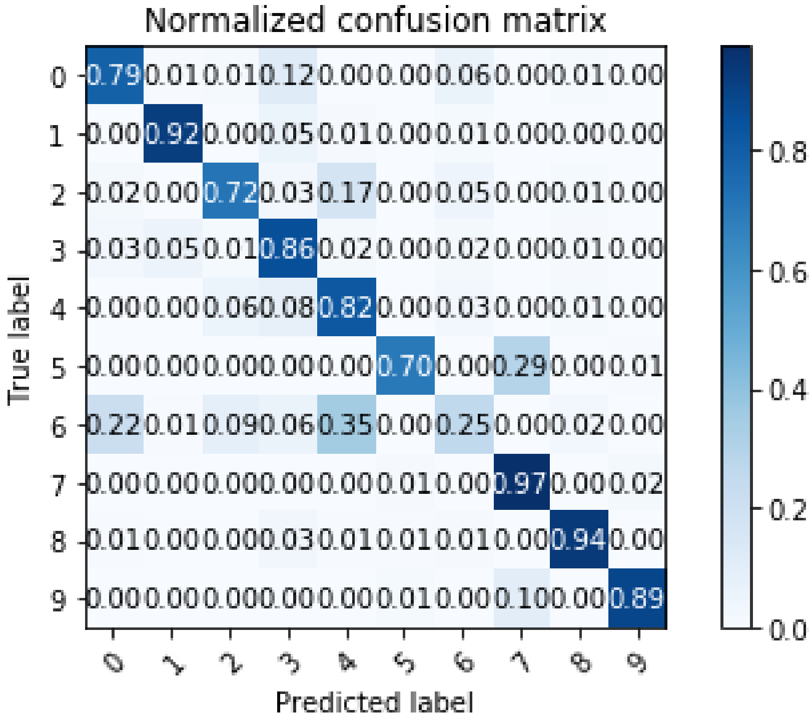


Figure 2

Figure 2 is showing the confusion matrix that is normalized this means that every row is divided by the sum of all the data's in that row

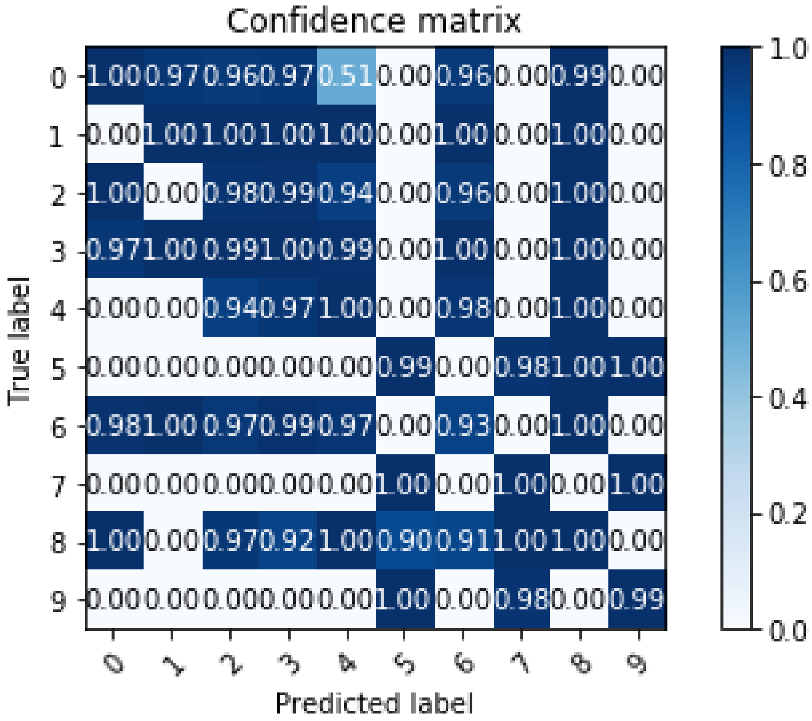


Figure 3

Figure 3 is the confidence matrix of this classification we can see that the data that are classified correct have a high confidence and that is a good thing but if some case are classified wrong they shouldn't high confidence but what we see in the matrix is the opposite and that is not a good thing the reason may be because of the probability functions may have a low covariance and that makes them quite different from each other and one probability might be much bigger than the other.

Question 4

We are now going to use Risk in our classifications and when the probability is above a certain point then we can pass the decision this means that we have a new class called the $(c+1)$ class and if the price of not making any decision is less than making a decision with a probability of being wrong. The point is to minimize the Risk of the classification

$$R(a_i|x) = \sum_{j}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|x) = \lambda_s \sum_{j=1,i\neq j}^{c} P(\omega_j|x) = \lambda_s\left(1 - P(\omega_j|x)\right) \rightarrow$$

$R(\alpha_i|x)$ is the total risk and what we are going to do is to minimize the risk of the classification and we need to choose either to make a decision or not to make one and the rick of not making a choice is λ_r and we need to compare this two, the one

$\lambda_r\ and\ \lambda_s\left(1 - P(\omega_j|x)\right)$ if for every number of j between 1 to C if λ_r was smaller than all amount of $\lambda_s\left(1 - P(\omega_j|x)\right)$ then not making any decision was lower rick.

$\forall i = 1 \dots C \quad \lambda_r < \lambda_s(1 - P(\omega_i|x))$ then we are going to label it as label 10.

And if $\exists i = 1 \dots C\ \ \lambda_r > \lambda(1 - P(\omega_i|x))$ then it's better to make a decision.

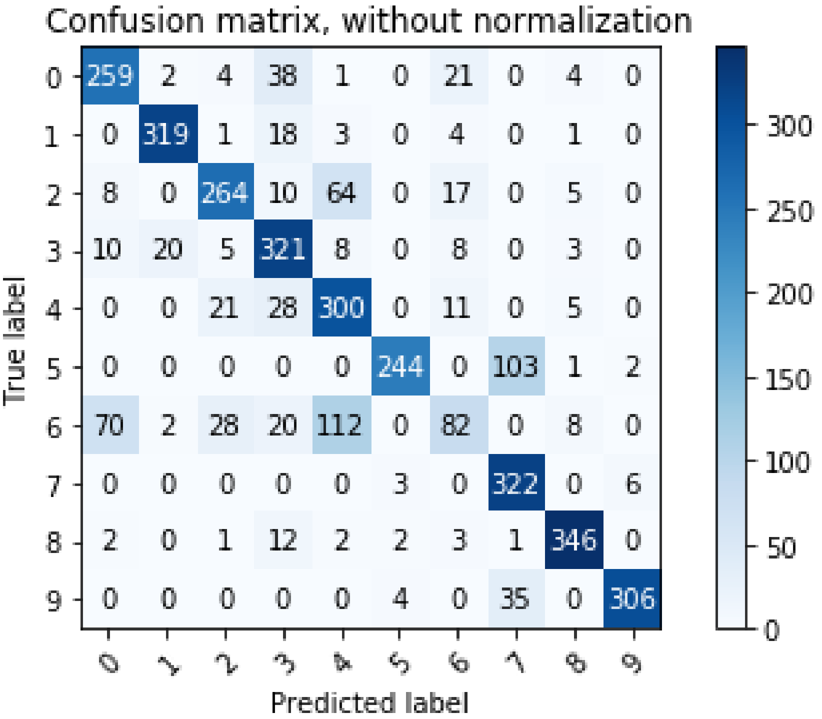
$$g_i(x) = \begin{cases} p(x|\omega_i)P(\omega_i) \quad i = 1,2,\dots,C \\ \left(1 - \dfrac{\lambda_r}{\lambda_s}\right)\displaystyle\sum_{j=1}^{C} p(x|\omega_i)P(\omega_i) \quad otherwise \end{cases}$$

The following result is from the code that we have programmed.

Time of Train: 1.56875

Time of Test: 138.1875

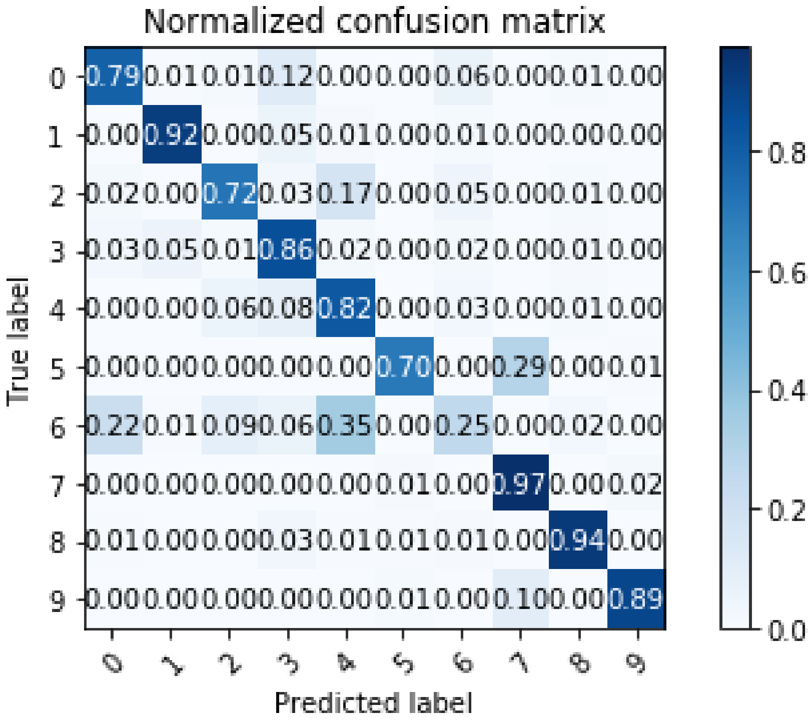
Accuracy: 78.94%


Confusion matrix, without normalization

	0	1	2	3	4	5	6	7	8	9
0	259	2	4	38	1	0	21	0	4	0
1	0	319	1	18	3	0	4	0	1	0
2	8	0	264	10	64	0	17	0	5	0
3	10	20	5	321	8	0	8	0	3	0
4	0	0	21	28	300	0	11	0	5	0
5	0	0	0	0	0	244	0	103	1	2
6	70	2	28	20	112	0	82	0	8	0
7	0	0	0	0	0	3	0	322	0	6
8	2	0	1	12	2	2	3	1	346	0
9	0	0	0	0	0	4	0	35	0	306

Figure 4 $\lambda_s = 1$ $\lambda_r = 0.8$

Figure 3 shows the confusion matrix without normalization


Normalized confusion matrix

	0	1	2	3	4	5	6	7	8	9
0	0.79	0.01	0.01	0.12	0.00	0.00	0.06	0.00	0.01	0.00
1	0.00	0.92	0.00	0.05	0.01	0.00	0.01	0.00	0.00	0.00
2	0.02	0.00	0.72	0.03	0.17	0.00	0.05	0.00	0.01	0.00
3	0.03	0.05	0.01	0.86	0.02	0.00	0.02	0.00	0.01	0.00
4	0.00	0.00	0.06	0.08	0.82	0.00	0.03	0.00	0.01	0.00
5	0.00	0.00	0.00	0.00	0.00	0.70	0.00	0.29	0.00	0.01
6	0.22	0.01	0.09	0.06	0.35	0.00	0.25	0.00	0.02	0.00
7	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.97	0.00	0.02
8	0.01	0.00	0.00	0.03	0.01	0.01	0.01	0.00	0.94	0.00
9	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.10	0.00	0.89

Figure 5 $\lambda_s = 1$ $\lambda_r = 0.8$

Figure 4 shows the confusion matrix with normalization.

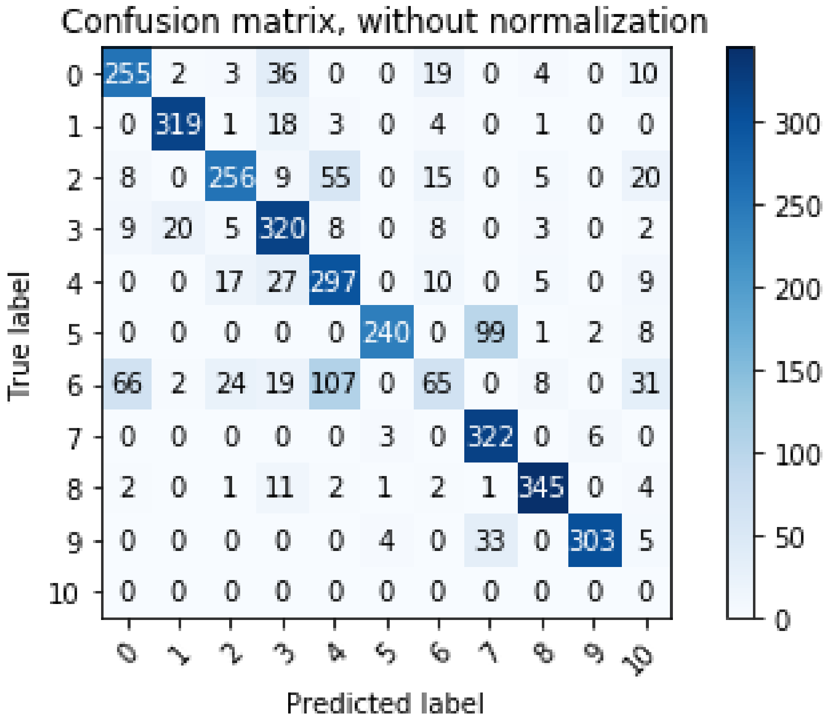
We can see from figure 3 and 4 that there were no case that we can reject the reason is that the difference of two probability is quit big so if we want to see that condition we changed the number of λ .

In the following result we can see some cases that the data were in class 10

Time of Train: 1.484

Time of Test: 141.21

Accuracy: 77.77%



Confusion matrix, without normalization

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9	10
0	255	2	3	36	0	0	19	0	4	0	10
1	0	319	1	18	3	0	4	0	1	0	0
2	8	0	256	9	55	0	15	0	5	0	20
3	9	20	5	320	8	0	8	0	3	0	2
4	0	0	17	27	297	0	10	0	5	0	9
5	0	0	0	0	0	240	0	99	1	2	8
6	66	2	24	19	107	0	65	0	8	0	31
7	0	0	0	0	0	3	0	322	0	6	0
8	2	0	1	11	2	1	2	1	345	0	4
9	0	0	0	0	0	4	0	33	0	303	5
10	0	0	0	0	0	0	0	0	0	0	0

Figure 6 $\lambda_s = 10$ $\lambda_r = 0.8$

Figure 5 shows the confusion matrix without normalization. In this part the defined $\lambda_s =$ 10 and that is quite different from the past one we can see different cases that have went is label 10

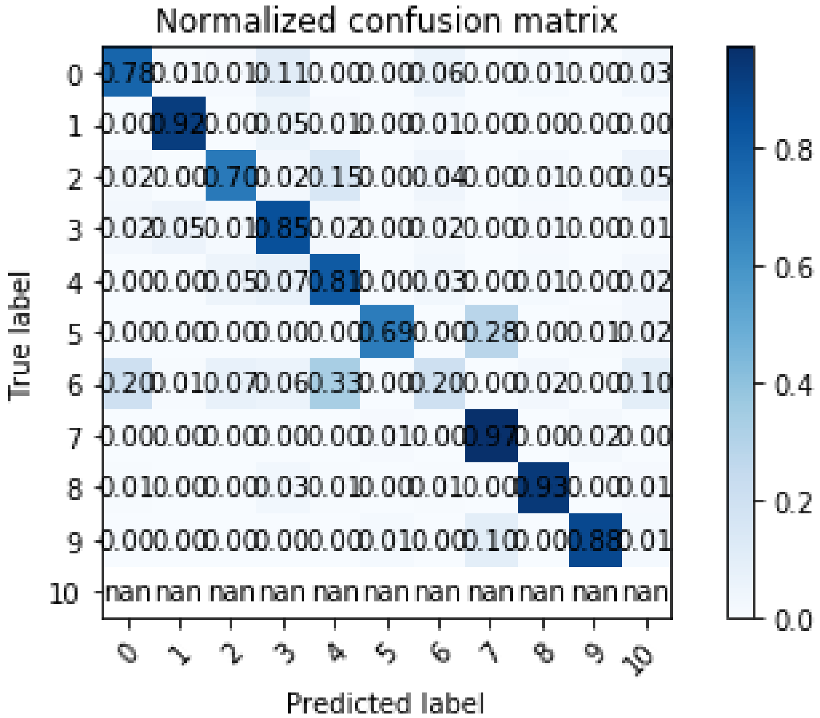


Figure 7 $\boldsymbol{\lambda_s = 10}$ $\boldsymbol{\lambda_r = 0.8}$

Figure show the normalized confusion matrix.

There are 90 cases that were rejected from the classification.

Now we can see if the cost of rejecting a decision is less than the cost of a probability of getting a wrong answer then rejecting a case may be a better choice.

Question 5

In this part we are going some different ways to classify our data

I) The fist classifier is a Parzen classifier with two shapes (Rectangular and Gaussian). The estimation of Parzen window:

$$p_N(\underline{x}) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{V_N}\varphi\left(\frac{x - x_k}{h_n}\right)$$

We can have different type of window function

$\varphi(x) = \begin{cases} 1 & |x| \leq \frac{1}{2} \\ 0 & else \end{cases}$ is a rectangular function which means that if all the arrays are smaller than $\frac{1}{2}$ we can apply that the data is in the window

$$\varphi(x) = \frac{1}{\left(\sqrt{2\pi}\right)^d h_n^d}\exp\left(-\frac{1}{2}\left(\frac{x - x_i}{h_n}\right)^2\right)$$

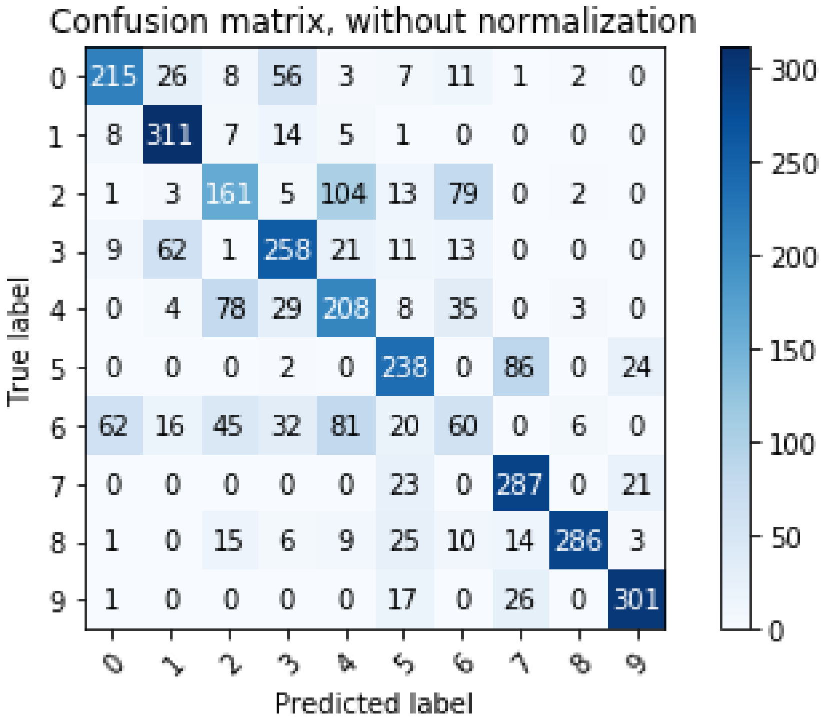
The following window function is a Gaussian kind.

At first we normalized our data by multiplying it in 0.01 and after that we done the rest of the computations.

Time computing: 950.578125

Accuracy: 66.42%

In this case we have determined the $h_n = 16$ and we got the result that is mentioned.



	0	1	2	3	4	5	6	7	8	9
0	215	26	8	56	3	7	11	1	2	0
1	8	311	7	14	5	1	0	0	0	0
2	1	3	161	5	104	13	79	0	2	0
3	9	62	1	258	21	11	13	0	0	0
4	0	4	78	29	208	8	35	0	3	0
5	0	0	0	2	0	238	0	86	0	24
6	62	16	45	32	81	20	60	0	6	0
7	0	0	0	0	0	23	0	287	0	21
8	1	0	15	6	9	25	10	14	286	3
9	1	0	0	0	0	17	0	26	0	301

Figure 8 Parzen Rectangular hn=16

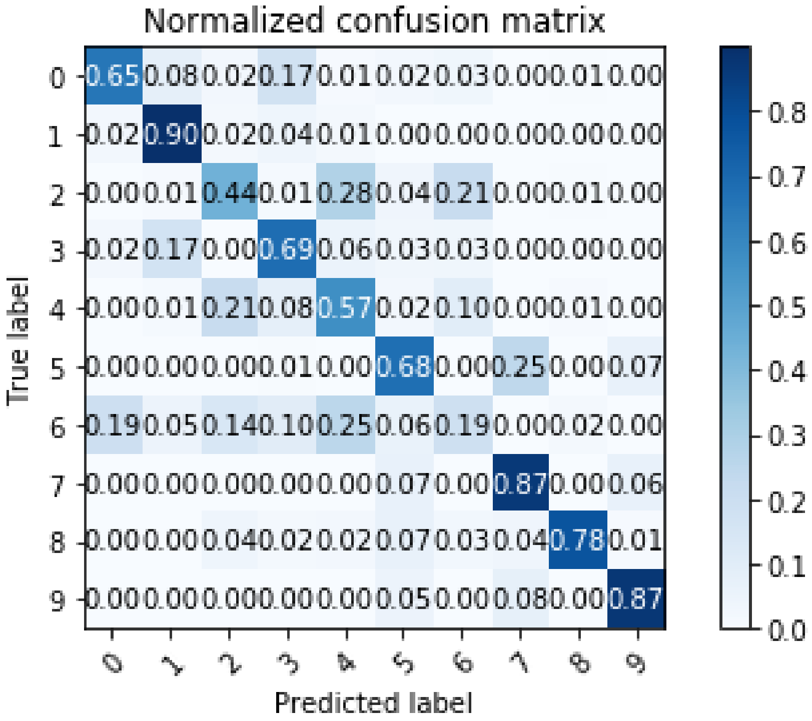


Figure 9 Parzen Rectangular hn=16

We may now change the value of h_n to see the different result. In this new classification we have determined the $h_n = 40$ and the result is like this.

Time computing: 1074.421875

Accuracy: 33.74%

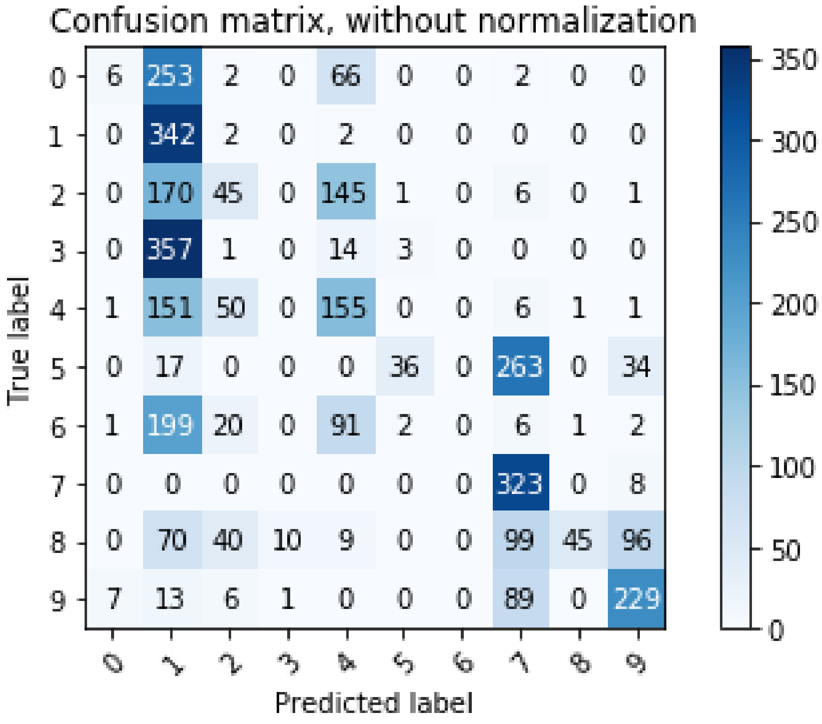


Figure 10 $h_n = 40$ Parzen Rectangular

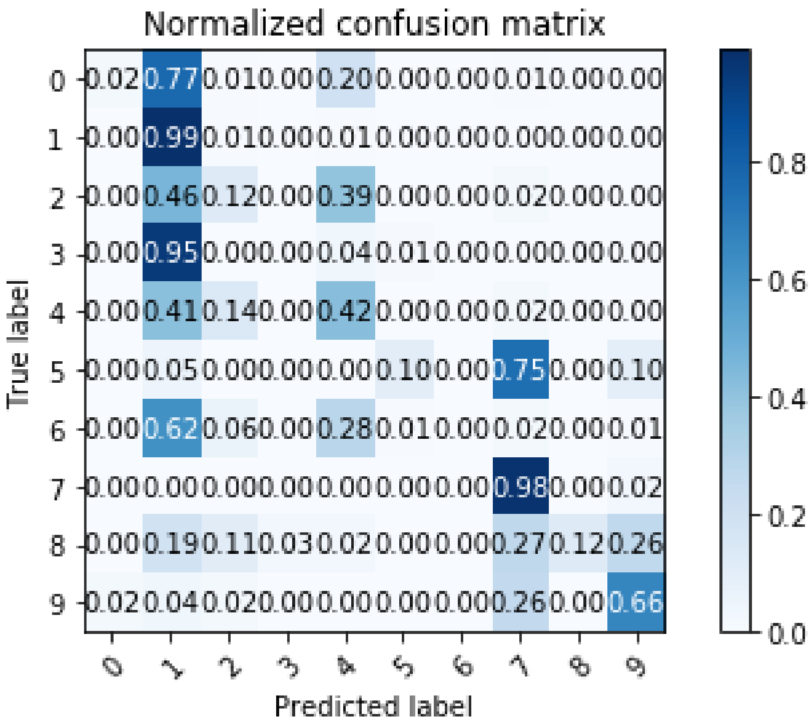


Figure 11

We can see that the h_n can't assume any number to h_n if we want to get a good accuracy there is a range that will be suitable for the classification if the h_n is to big the dependency to that point will decrease and if we choose the h_n quite small then the data will stick to the train data and would not be suitable for generalizing the classifier

For example if we set $h_n = 8$ we will get this as the result.

Time of computing: 997.39

Accuracy: 76.68%

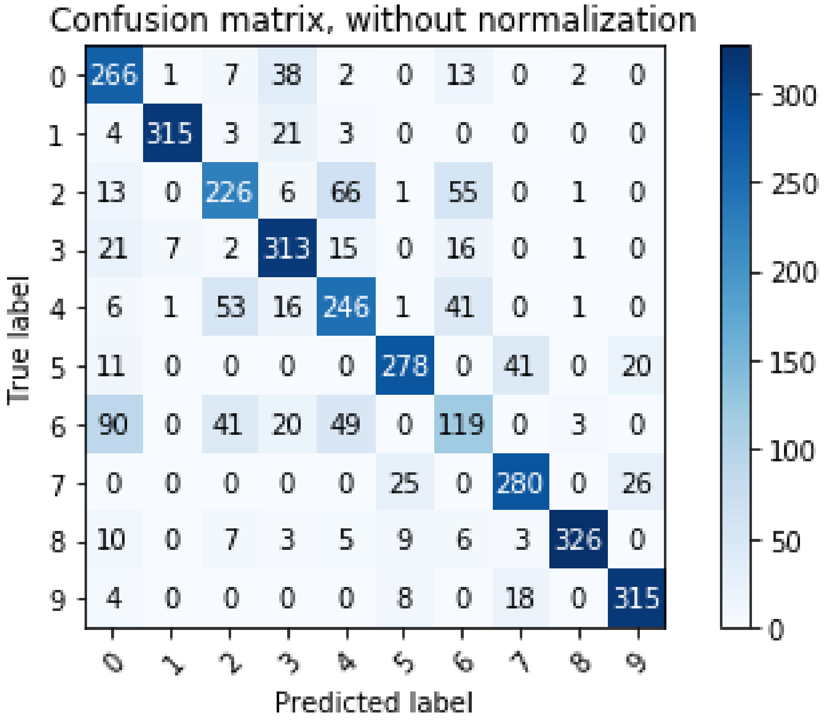


Figure 12 $h_n = 8$

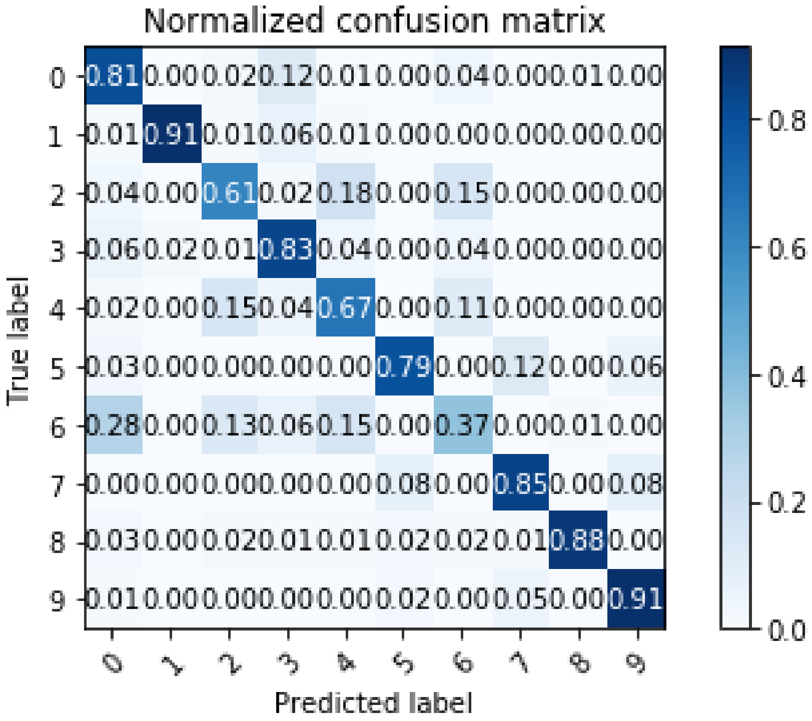


Figure 13

Gaussian Parzen window:

In this calculation set that we have done we determined the $h_n = 2\sqrt{2}$

Time computing: 446.0625

Accuracy: 66.37 %

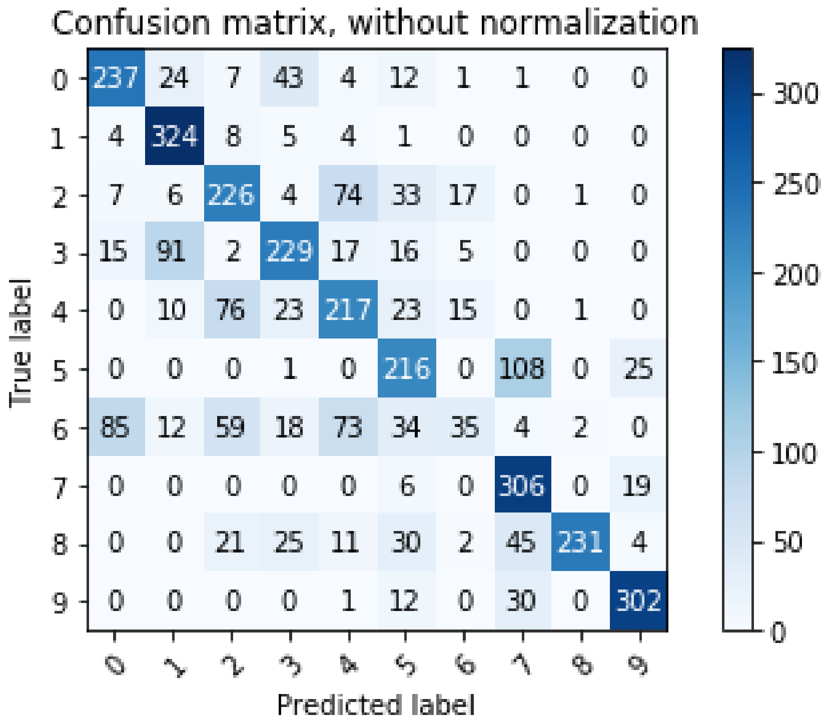


Figure 14 Gaussian

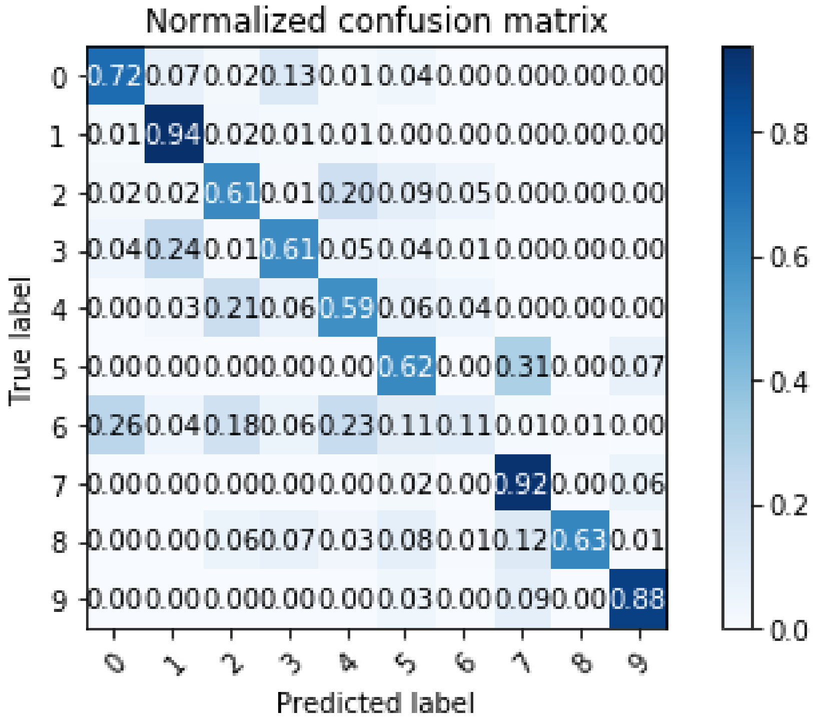


Figure 15

II) In this part we are going to use another algorithm called (k-NN) non-parametric estimate of pdf. In this case $k = 10$

Time computing: 324.734375

Accuracy: 81.25%

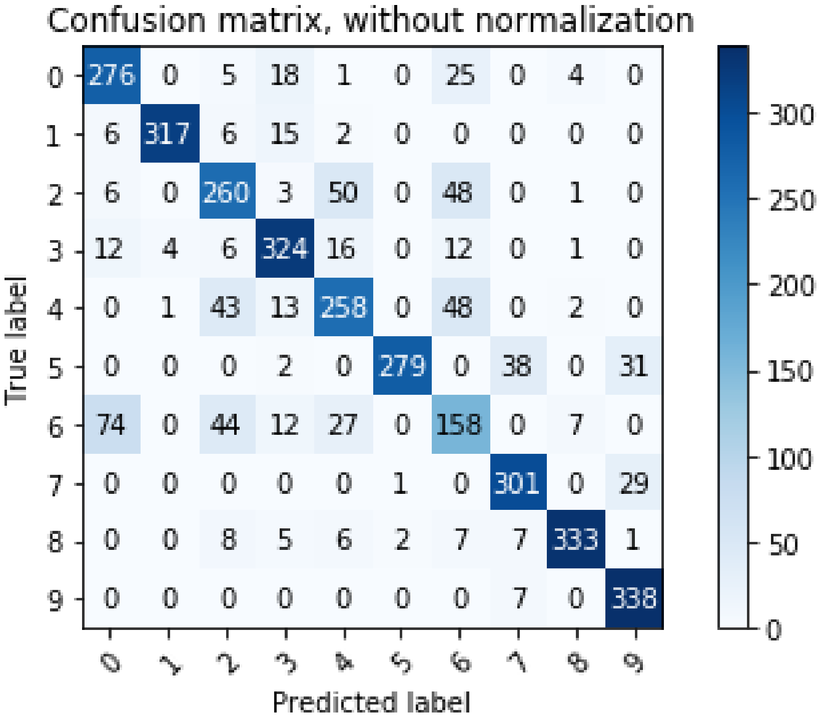


Figure 16

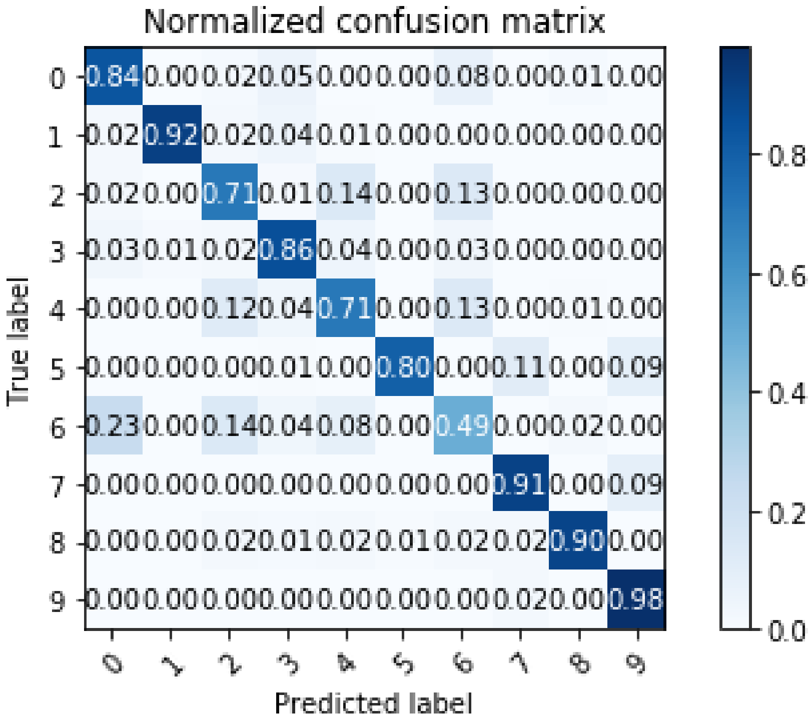


Figure 17

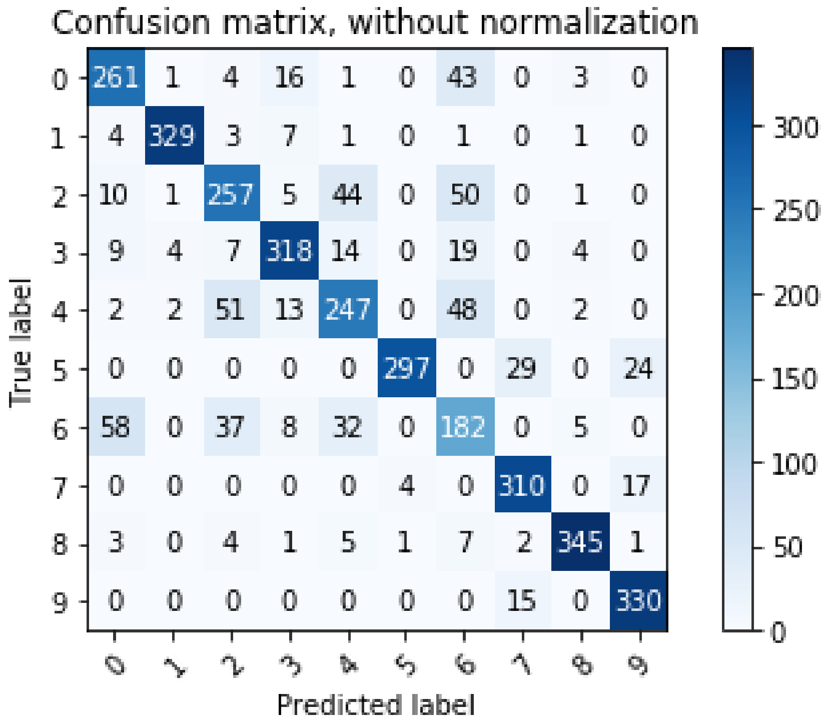
We change the k and see the result it is reasonable that if the value of k is too big or too small the result will have low accuracy

III) In this method we are going to design a classifier that will use k-nearest-neighbor algorithm to classify the dataset that we have.

For $k = 1$

Time computing: 686.03125

Accuracy: 82.17%



	0	1	2	3	4	5	6	7	8	9
0	261	1	4	16	1	0	43	0	3	0
1	4	329	3	7	1	0	1	0	1	0
2	10	1	257	5	44	0	50	0	1	0
3	9	4	7	318	14	0	19	0	4	0
4	2	2	51	13	247	0	48	0	2	0
5	0	0	0	0	0	297	0	29	0	24
6	58	0	37	8	32	0	182	0	5	0
7	0	0	0	0	0	4	0	310	0	17
8	3	0	4	1	5	1	7	2	345	1
9	0	0	0	0	0	0	0	15	0	330

Figure 18

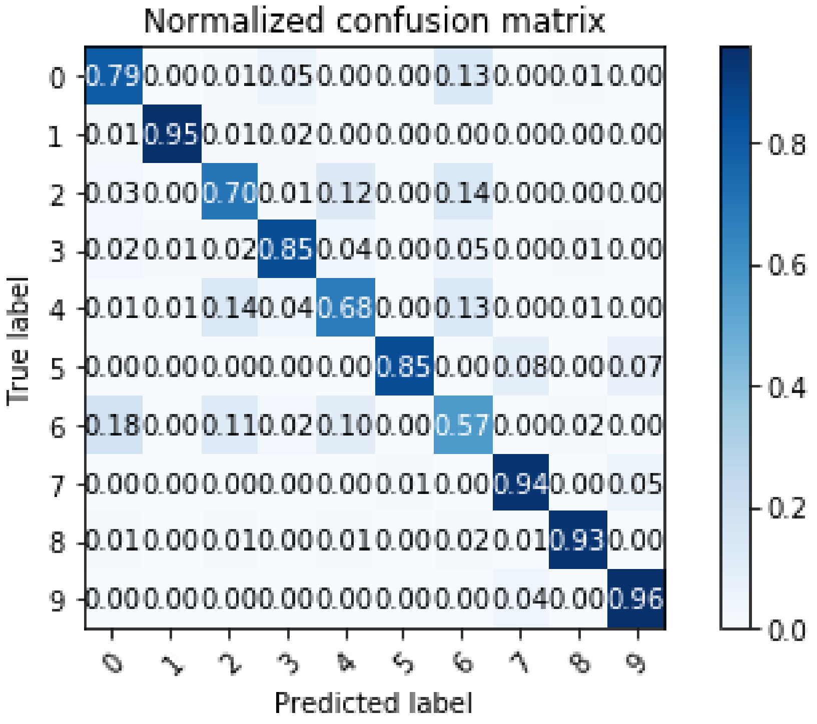


Figure 19

For $k = 3$

Time computing: 1005.78125

Accuracy: 81.48%

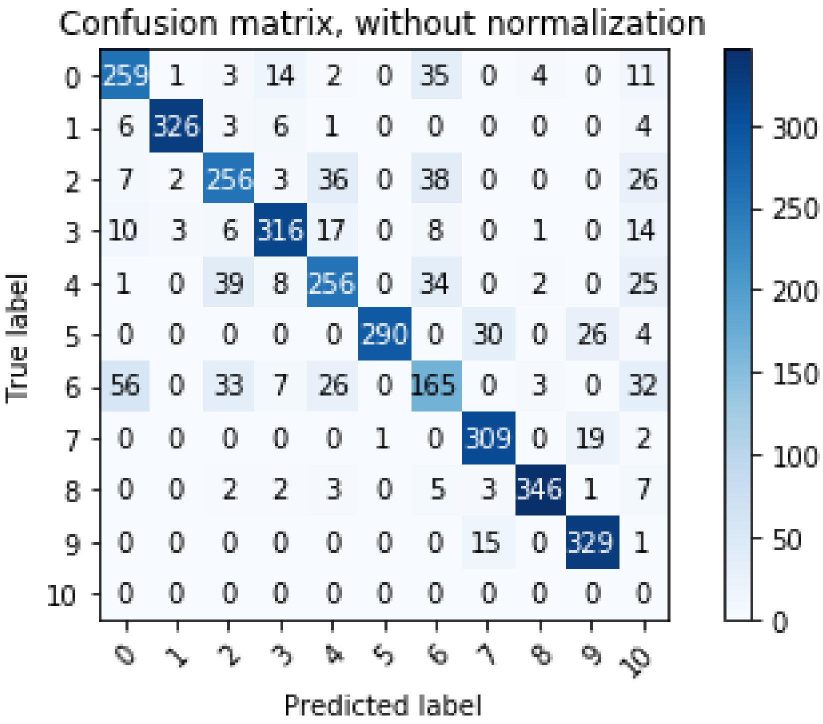


Figure 20

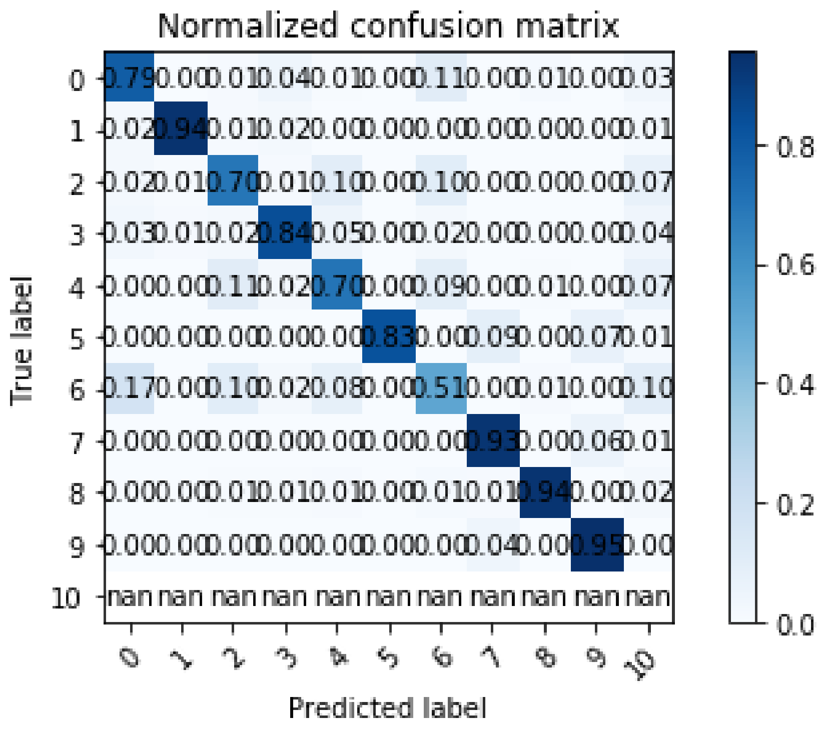


Figure 21

For $k = 5$

Time computing: 1392.87795984499996

Accuracy: 80.82%

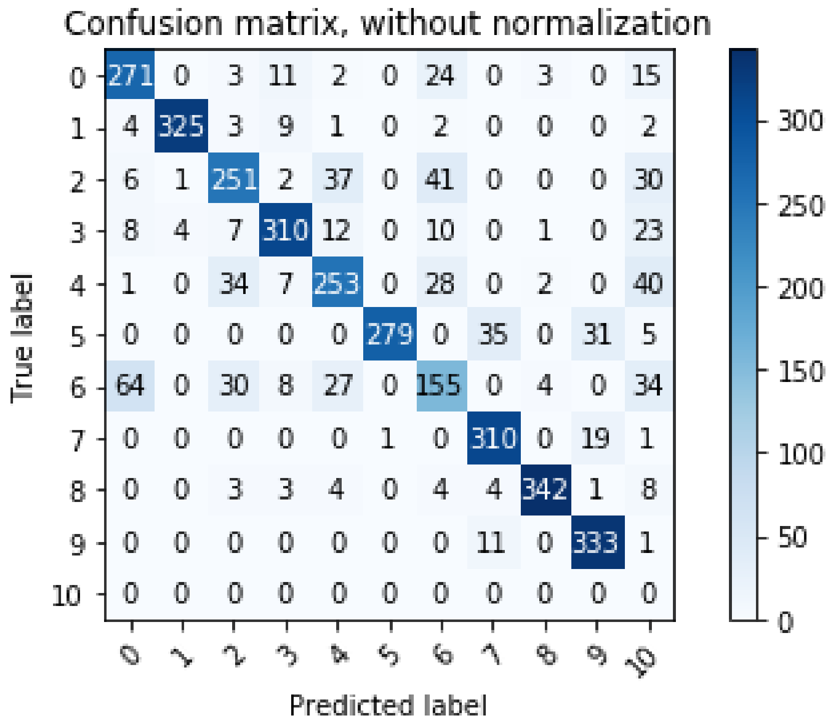


Figure 22

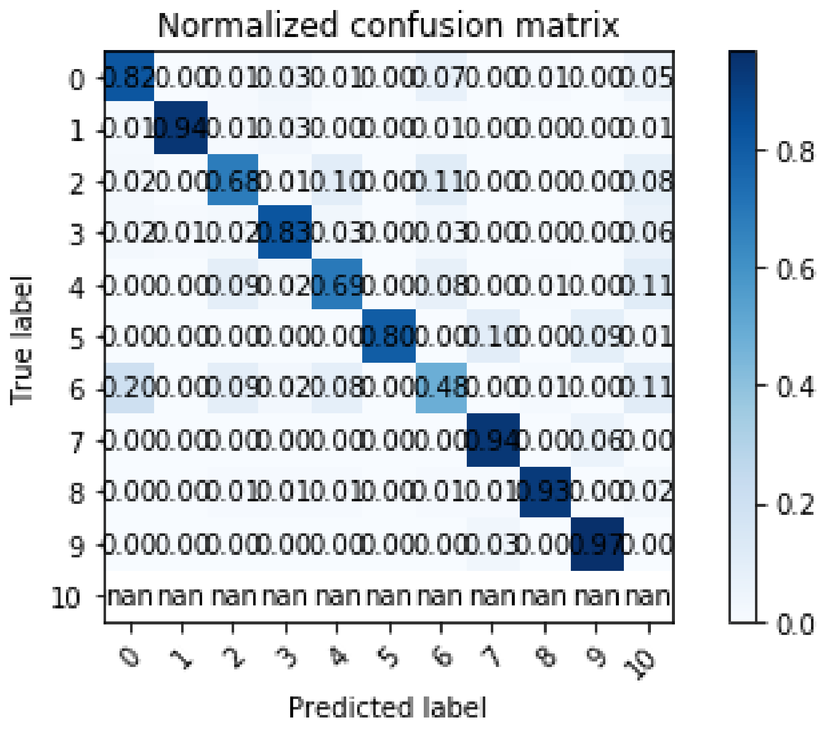


Figure 23

For $k = 10$

Time computing: 1937.25

Accuracy: 81.02%

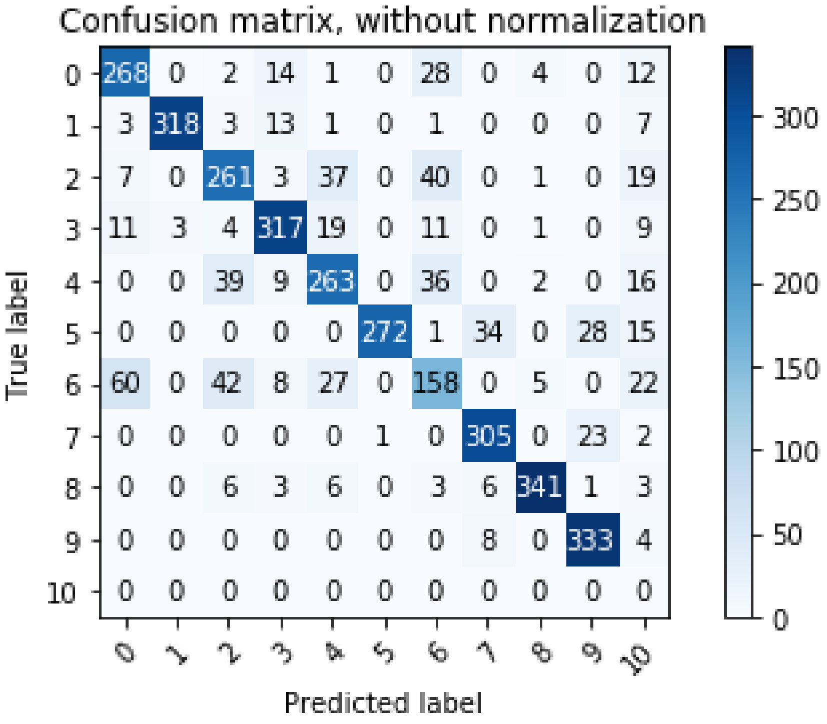


Figure 24

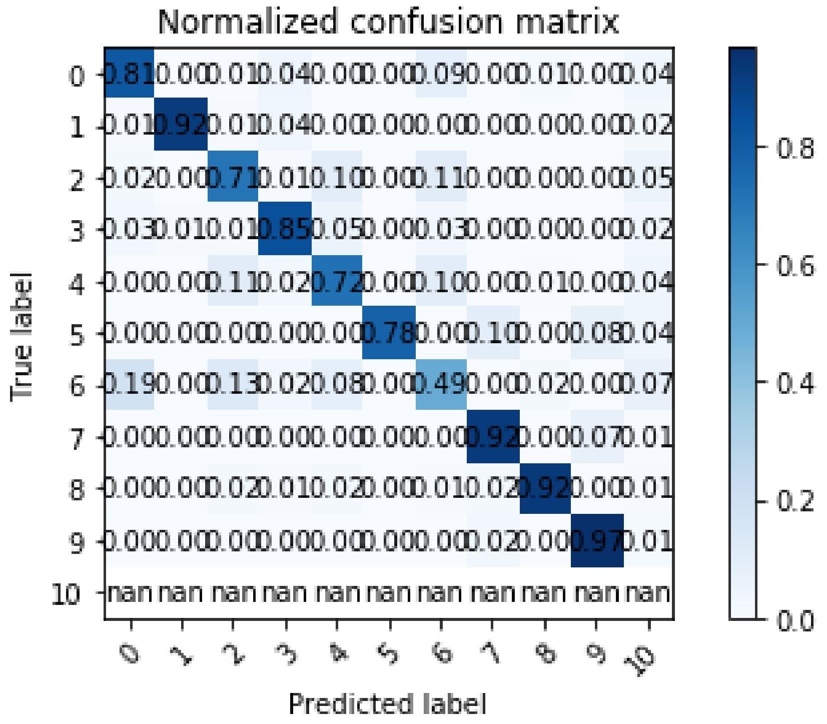


Figure 25

We saw from the accuracy and the correction rate that it has a goodness of fit and if it's too big or too small it will not give us a good answer.

IV) In this algorithm you need the define the mean distance from a class and the minimal average distance will be the predicted class.

$d_i(x) = \frac{1}{N_i}\sum_{q=1}^{N_i}\left\|\underline{x}-\underline{x}^q\right\|_k$ which N_i is the number of data that belong to class i and the result will be $i^* = argMin_i^C\left(d_i(x)\right)$

Time computing: 392.87795984499996

Accuracy: 64.62%

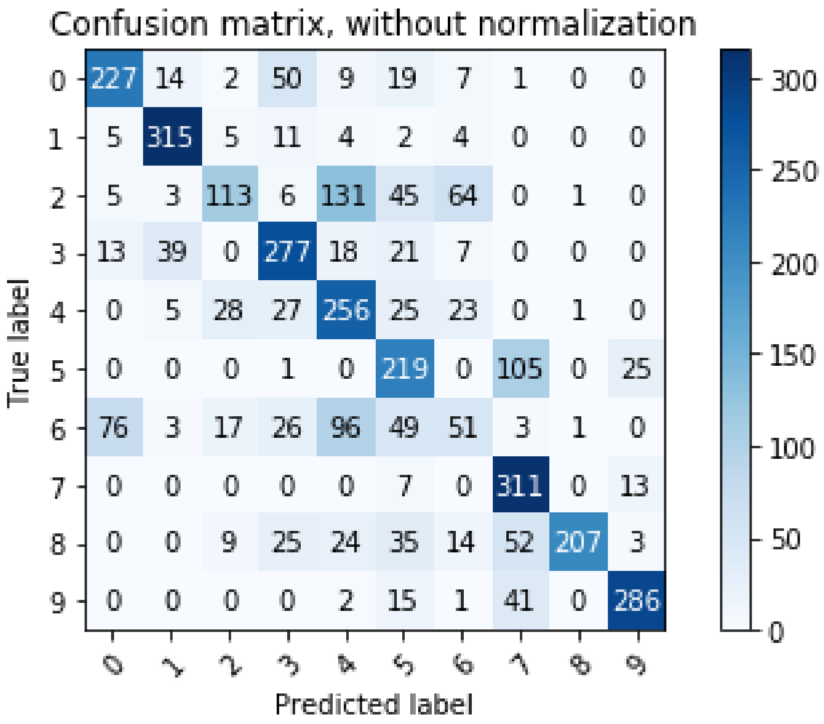


Figure 26

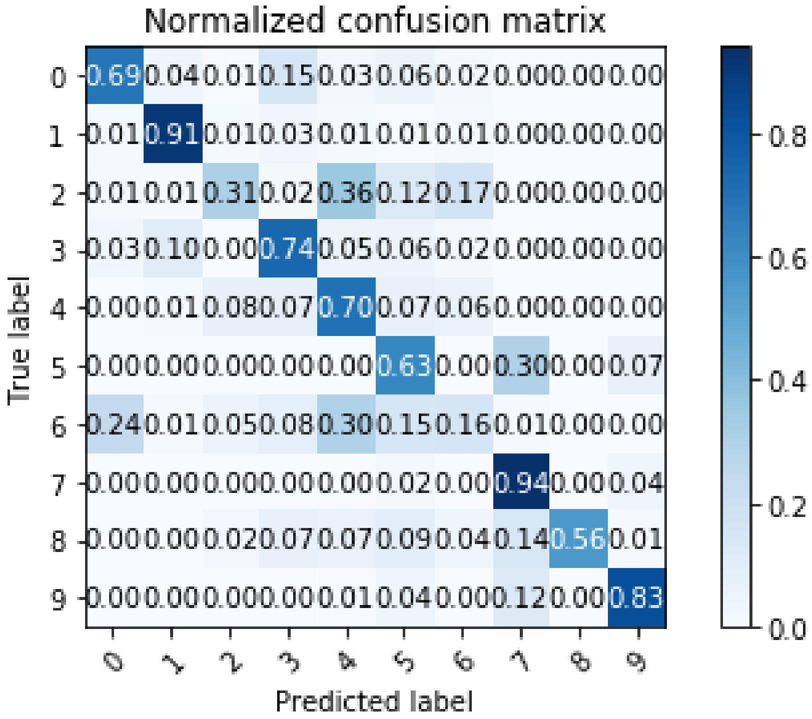


Figure 27

Question 6

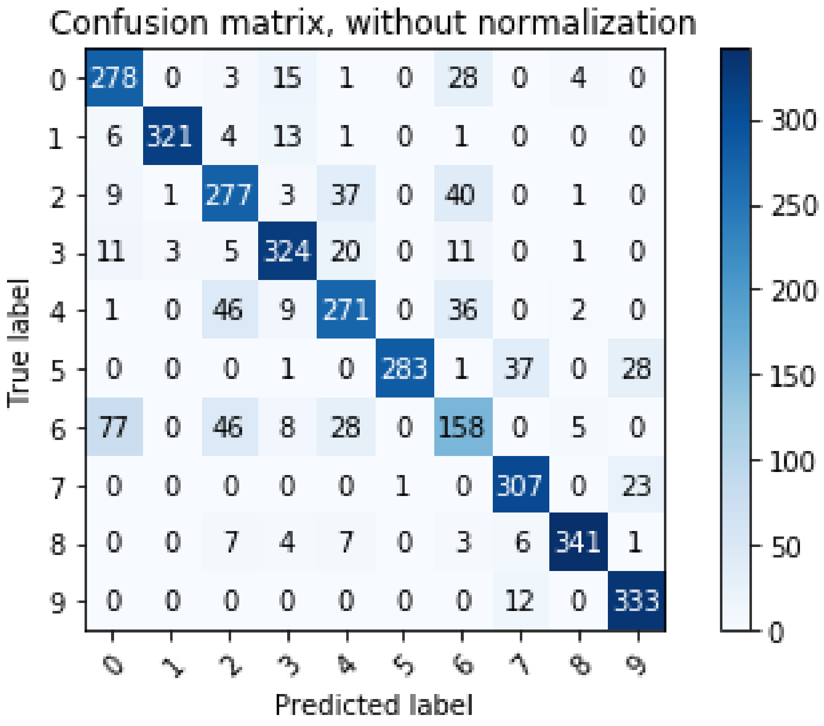
Question 6.1

I) For KNN classifier we have this result for $k = 10$

Time of train: 0.1093

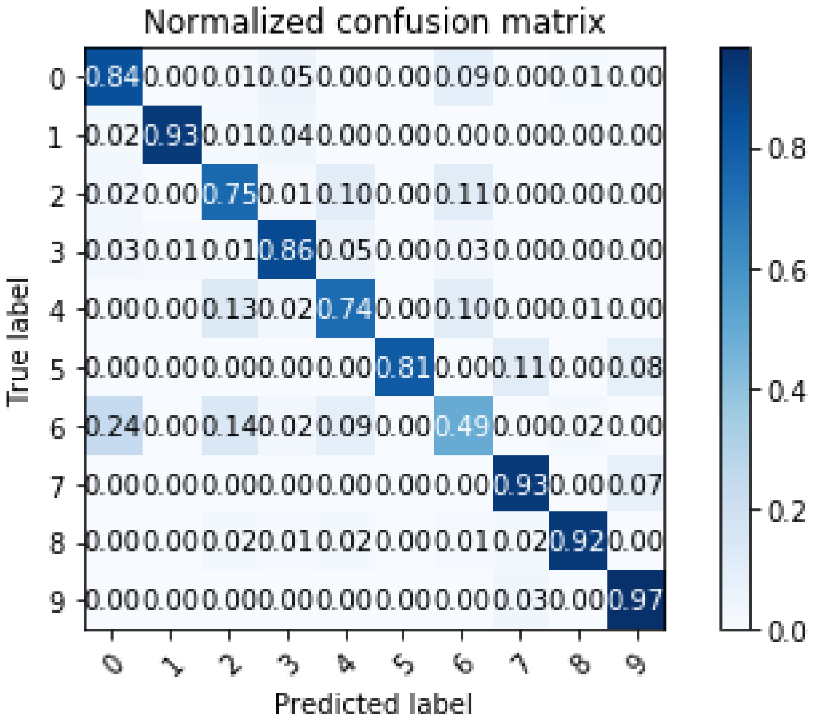
Time of test: 5.0

Accuracy: 82.26%



True label	0	1	2	3	4	5	6	7	8	9
0	278	0	3	15	1	0	28	0	4	0
1	6	321	4	13	1	0	1	0	0	0
2	9	1	277	3	37	0	40	0	1	0
3	11	3	5	324	20	0	11	0	1	0
4	1	0	46	9	271	0	36	0	2	0
5	0	0	0	1	0	283	1	37	0	28
6	77	0	46	8	28	0	158	0	5	0
7	0	0	0	0	0	1	0	307	0	23
8	0	0	7	4	7	0	3	6	341	1
9	0	0	0	0	0	0	0	12	0	333

Figure 28



True label	0	1	2	3	4	5	6	7	8	9
0	0.84	0.00	0.01	0.05	0.00	0.00	0.09	0.00	0.01	0.00
1	0.02	0.93	0.01	0.04	0.00	0.00	0.00	0.00	0.00	0.00
2	0.02	0.00	0.75	0.01	0.10	0.00	0.11	0.00	0.00	0.00
3	0.03	0.01	0.01	0.86	0.05	0.00	0.03	0.00	0.00	0.00
4	0.00	0.00	0.13	0.02	0.74	0.00	0.10	0.00	0.01	0.00
5	0.00	0.00	0.00	0.00	0.00	0.81	0.00	0.11	0.00	0.08
6	0.24	0.00	0.14	0.02	0.09	0.00	0.49	0.00	0.02	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.07
8	0.00	0.00	0.02	0.01	0.02	0.00	0.01	0.02	0.92	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.97

Figure 29

II) Parzen non-parametric estimation

For $r = 2028$

Time of Train: 0.625

Time of Test: 25.70

Accuracy:64.82%

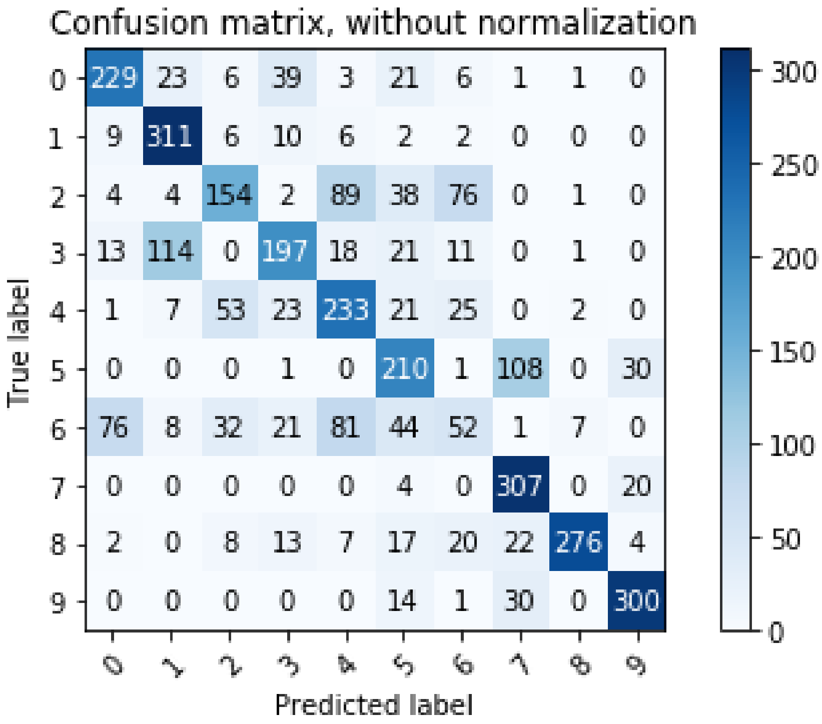


Figure 30

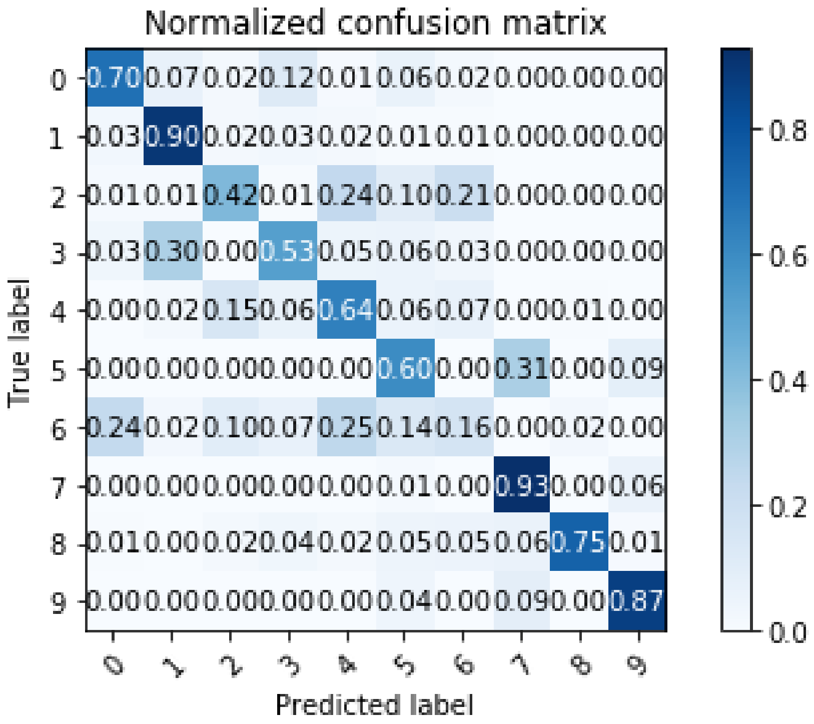


Figure 31

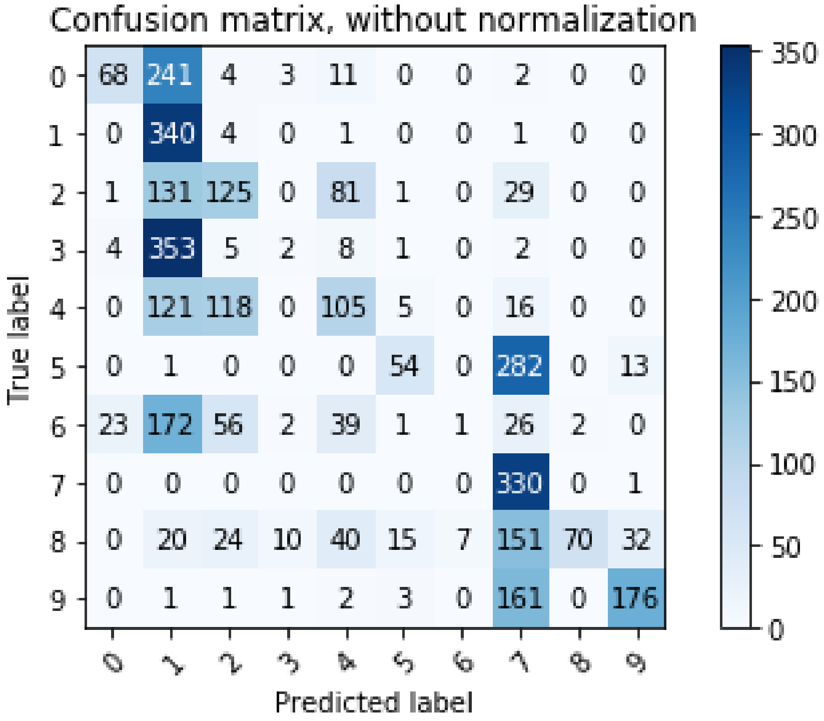
For $r = 1500$ →no neighbor has been found

For $r = 3000$

Time of Train: 0.5625

Time of Test: 34.26

Accuracy: 36.21%



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	68	241	4	3	11	0	0	2	0	0
1	0	340	4	0	1	0	0	1	0	0
2	1	131	125	0	81	1	0	29	0	0
3	4	353	5	2	8	1	0	2	0	0
4	0	121	118	0	105	5	0	16	0	0
5	0	1	0	0	0	54	0	282	0	13
6	23	172	56	2	39	1	1	26	2	0
7	0	0	0	0	0	0	0	330	0	1
8	0	20	24	10	40	15	7	151	70	32
9	0	1	1	1	2	3	0	161	0	176

Figure 32

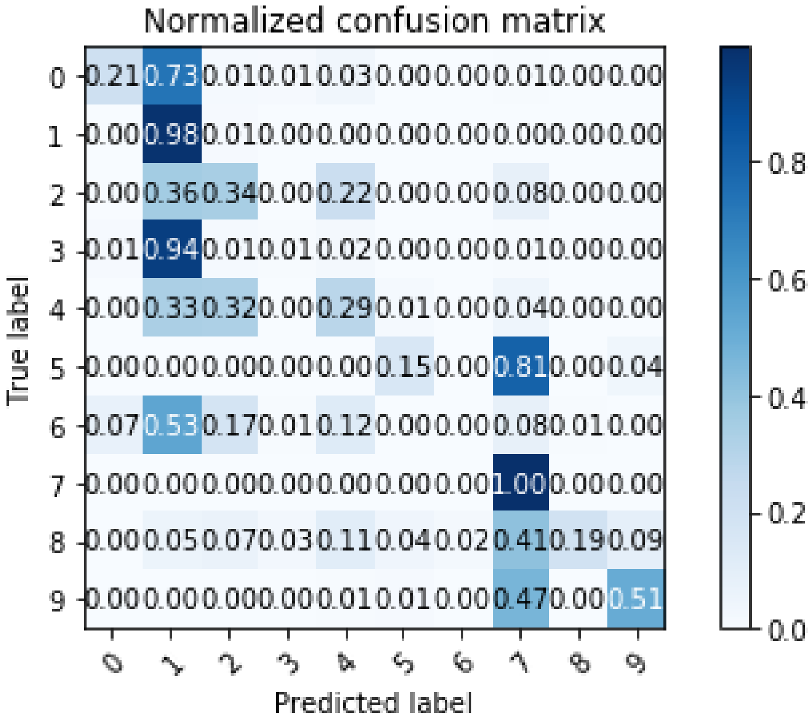


Figure 33

We can see that if we choose a r that either is too big of too small we have low accuracy in the classification.

III) Gaussian Naïve Bayes

Time of Train: 0.4065

Time of Test: 0.531

Accuracy: 73.37%

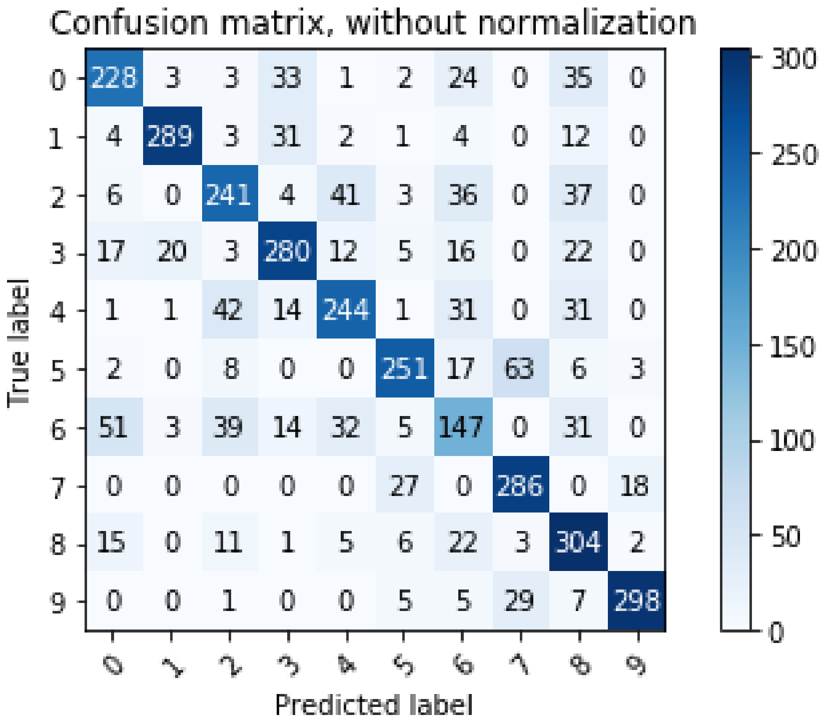


Figure 34

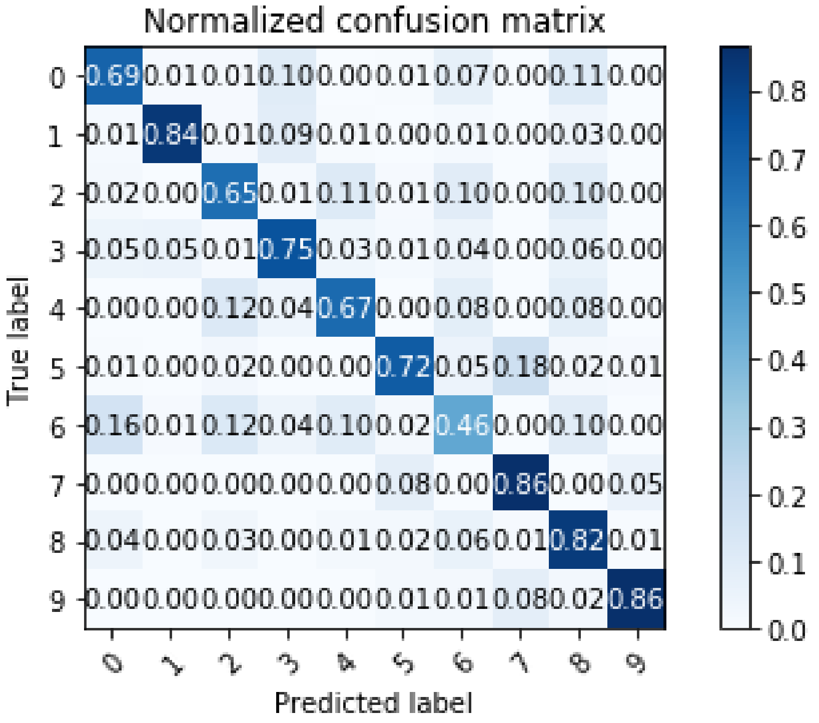


Figure 35

Question 6.2

I) In this part we are going to compare the classifiers in question 3,4 and 6

In some cases, in the algorithm that we have implemented there is no time as the train time because we are computing the distance or the number of data that are in the window

Classifier name	Accuracy	Confusion matrix	Confidence matrix	Time of Train	Time of Test
Bayes optimal	78.94%	good	Not bad	1.64	126.51
Parzen Rectangular $h_n = 16$	66.42%	Not bad		-	950.57
Parzen Rectangular $h_n = 40$	33.74%	bad		-	1074.42
Parzen Rectangular $h_n = 8$	76.68%	good		-	997.39
Parzen Gaussian $h_n = 2\sqrt{2}$	66.37%	Not bad		-	446.06
KNN $k = 10$	81.25%	good		-	324.73
k-nearest-neighbor $k = 1$	81.17%	good		-	686.03
k-nearest-neighbor $k = 3$	81.48%	good		-	1005.78
k-nearest-neighbor $k = 5$	80.82%	good		-	1392.87
k-nearest-neighbor $k = 10$	81.02%	good		-	1937.25
List mean distance	64.62%	Not bad		-	392.87

Skylearn KNN $k = 10$	82.26%	good	-	0.1093	5
Skylearn Parzen $r = 2028$	64.82%	Not bad	-	0.625	25.70
Skylean Parzen $r = 1500$	-	-	-	-	-
Skylearn Parzen $r = 3000$	36.21%	bad	-	0.5625	34.26
Gaussian Native Bayes	73.37%	good		0.4065	0.531

We can see from the table that the non-parametric classifier has a better accuracy but will take a longer time to give us a result also we can understand from the confusion matrix that higher accuracy brings a better confusion matrix and KNN and k-nearest-neighbor are the best classifiers from this CA.

For me KNN will be the best classifier because for me accuracy is the most important thing although time is an important thing but in my case I thing accuracy I quite important and in some cases classifier that have low accuracy take long time to give result.

Process

In this Homework we are using different methods to classify our data set and we also used confusion and confidence matrix to give us a better view of the classification. We also compared the result when we change the parameters of the classification and when we changed the type of the classifier that we have

Reference

Automatic speech recognition a deep learning approach

stackoverflow.com

youtube.com

https://scholar.google.com/scholar?q=Forward+Algorithm+using+hidden+markov+model&hl=fa&as_sdt=0&as_vis=1&oi=scholart

https://eu.udacity.com/course/introduction-to-computer-vision--ud810

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.RadiusNeighborsClassifier.html#sklearn.neighbors.RadiusNeighborsClassifier

https://sebastianraschka.com/Articles/2014_kernel_density_est.html