Part 1

1 Dataset

```
A Random member of each datasets is as follow:
```

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11 …	Y
Train	+1.29	+0.64	+2.43	+2.51	+3.39	+1.45	+1.87	+1.41	+0.50	+2.82	+2.88	-
Valid	+2.64	+4.02	+4.65	+3.97	+3.83	+3.45	+2.63	+1.17	+3.40	+1.68	+3.20	+
Test	+4.54	+3.08	+4.49	+4.13	+3.25	+5.26	+2.27	+2.48	+3.59	+5.42	+2.94	-

2.a For DS1, report the best fit accuracy achieved by the classifier.

I used the Training set as the question asked. However, because the problem has no hyperparameter to learn, we could use the Validation set alongside the Training set to get better accuracy.

```
The accuracy for the test set is: 94.38%
```

2.b Report the coefficients learned.

W_0 = +26.79

	1	2	3	4	5	6	7	8	9	10
W :	+13.97	-8.29	-5.69	-3.22	-9.48	-3.99	+16.56	-23.30	-28.46	+8.90
μ_1:	+1.198	+1.199	+1.113	+1.214	+1.160	+1.163	+1.187	+1.245	+1.169	+1.17
μ_2:	+2.019	+2.008	+2.001	+2.028	+2.024	+1.973	+2.057	+1.976	+2.038	+1.996

	11	12	13	14	15	16	17	18	19	20
W :	-12.73	-11.90	+15.14	+12.72	-5.37	+12.58	+28.67	-6.63	-0.86	-4.80
μ_1:	+1.228	+1.233	+1.211	+1.227	+1.214	+1.200	+1.150	+1.197	+1.249	+1.275
μ_2:	+2.039	+2.001	+2.040	+2.018	+2.065	+2.016	+1.986	+1.968	+2.042	+2.079

π_1 = 0.5 , π_1 = 0.5

Sigma :

7.61	5.39	5.8	4.97	5.58	5.86	4.57	5.24	4.81	5.02	3.79	5.06	6.77	5.85	5.83	5.8	5.58	5.42	5.51	5.8
5.39	6.95	5.17	4.25	5.35	5.49	4.32	3.91	4.04	5.03	3.32	4.56	5.81	5.02	5.28	5.11	5.54	5.	5.39	5.32
5.8	5.17	6.89	4.65	5.45	6.25	4.43	4.66	4.57	4.85	3.08	4.42	6.02	4.91	5.78	5.81	5.87	4.68	4.53	4.89
4.97	4.25	4.65	5.57	5.09	4.34	3.63	4.21	3.27	4.06	2.66	4.	5.61	4.68	4.63	4.93	4.5	4.32	3.87	5.67
5.58	5.35	5.45	5.09	6.8	5.16	4.89	4.25	4.5	5.01	4.	4.83	5.93	5.77	5.74	6.06	5.71	4.96	5.33	5.61
5.86	5.49	6.25	4.34	5.16	6.47	4.34	4.79	4.49	5.21	2.85	4.67	6.17	4.79	5.54	5.84	5.65	4.79	4.7	5.09
4.57	4.32	4.43	3.63	4.89	4.34	5.13	3.84	3.85	4.24	2.95	4.27	4.56	4.11	4.59	5.17	4.21	3.55	4.64	3.86
5.24	3.91	4.66	4.21	4.25	4.79	3.84	5.89	3.45	4.78	2.43	4.75	5.98	5.1	4.78	6.07	4.37	4.75	4.81	4.4
4.81	4.04	4.57	3.27	4.5	4.49	3.85	3.45	4.78	4.51	3.06	3.93	4.92	4.51	4.72	4.99	4.89	3.89	4.16	4.03
5.02	5.03	4.85	4.06	5.01	5.21	4.24	4.78	4.51	7.07	3.22	4.62	6.74	5.09	4.83	5.81	4.6	5.2	4.52	5.27
3.79	3.32	3.08	2.66	4.	2.85	2.95	2.43	3.06	3.22	3.31	2.79	3.53	3.87	3.48	3.58	3.54	3.09	3.7	3.4
5.06	4.56	4.42	4.	4.83	4.67	4.27	4.75	3.93	4.62	2.79	5.13	5.81	4.88	4.63	5.45	4.85	4.69	4.99	4.44
6.77	5.81	6.02	5.61	5.93	6.17	4.56	5.98	4.92	6.74	3.53	5.81	8.82	6.08	5.98	6.41	5.74	6.13	5.77	7.16
5.85	5.02	4.91	4.68	5.77	4.79	4.11	5.1	4.51	5.09	3.87	4.88	6.08	6.88	5.71	6.16	5.51	5.46	5.91	5.42
5.83	5.28	5.78	4.63	5.74	5.54	4.59	4.78	4.72	4.83	3.48	4.63	5.98	5.71	6.72	7.	5.6	4.86	5.38	5.59
5.8	5.11	5.81	4.93	6.06	5.84	5.17	6.07	4.99	5.81	3.58	5.45	6.41	6.16	7.	9.11	5.75	5.67	5.66	5.81
5.58	5.54	5.87	4.5	5.71	5.65	4.21	4.37	4.89	4.6	3.54	4.85	5.74	5.51	5.6	5.75	6.5	4.95	5.25	4.87
5.42	5.	4.68	4.32	4.96	4.79	3.55	4.75	3.89	5.2	3.09	4.69	6.13	5.46	4.86	5.67	4.95	5.75	4.77	5.03
5.51	5.39	4.53	3.87	5.33	4.7	4.64	4.81	4.16	4.52	3.7	4.99	5.77	5.91	5.38	5.66	5.25	4.77	6.44	4.92
5.8	5.32	4.89	5.67	5.61	5.09	3.86	4.4	4.03	5.27	3.4	4.44	7.16	5.42	5.59	5.81	4.87	5.03	4.92	7.93

3.a Does this classifier perform better than GDA or worse? Are there particular values of k which perform better? Why does this happen? Use validation accuracy for model selection.

Performs worse. It has just 62.12% performance, and it's lower than the accuracy of GDA (94.38%). Why? Because in the first method, we used a linear model for data generated linearly (k-NN is non-linear) and, as the number features are high (n=20), the model can be involved with a curse of dimensionality. This can lead to a high variance; as shown in Figure 1, the accuracy is not stable, and it can be solved by gathering more data. (Expected edge length = $0.5^{\frac{1}{20}} = 96.6\%$)

The best value for k is 233, which has the biggest accuracy in the Validation set. However, it's not a meaningful value for this parameter since the accuracy bounces in a short range of 58-62 % for an extended range of k.

```
kNN is ready!
Randomly chosen result:
       k :    3      82     152     488     527     587     610     657     844
Accuracy : 54.12%  59.62%  61.12%  59.38%  59.13%  59.50%  59.13%  58.88%  59.00%

Best Accuracy in Validation Set is 62.12% w.r.t. 233 Neighbors.
```

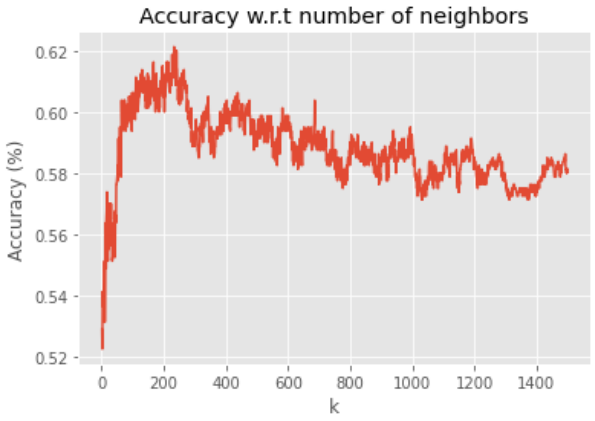

Figure 1

3.b Report the best fit accuracy achieved by this classifier.

```
kNN is ready!
Accuracy for Test Set is 56.50% w.r.t. 233 Neighbors.
```

4 Dataset

A Random member of each datasets is as follow:

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11 …	Y
Train	+2.44	-1.08	-2.40	+0.91	-3.45	+0.01	-0.94	-1.04	+0.85	+1.58	+1.40	+
Valid	+1.85	+0.89	+0.72	+0.76	+0.85	-3.13	+1.21	+3.22	-1.76	+1.13	-0.60	-
Test	+0.93	-0.52	+0.50	-1.74	+0.52	+0.11	-0.93	-0.53	+0.66	-1.16	-0.45	-

5.a For DS2, report the best fit accuracy achieved by the classifier.

The accuracy is: 52.0%

5.b Report the coefficients learned.

W_0 = +0.04

	1	2	3	4	5	6	7	8	9	10
W :	-0.01	-0.01	+0.00	-0.04	-0.02	+0.01	-0.02	-0.02	-0.00	+0.02
μ_1:	+1.072	+0.996	+0.928	+0.907	+0.986	+1.027	+0.934	+1.012	+0.993	+0.980
μ_2:	+1.235	+1.202	+1.201	+1.134	+1.163	+1.214	+1.223	+1.219	+1.191	+1.190

	11	12	13	14	15	16	17	18	19	20
W :	+0.02	-0.05	-0.03	-0.04	+0.02	+0.12	+0.03	+0.02	-0.03	-0.00
μ_1:	+1.020	+0.938	+0.989	+0.962	+0.987	+1.030	+1.014	+1.021	+0.952	+1.039
μ_2:	+1.227	+1.197	+1.168	+1.267	+1.193	+1.126	+1.208	+1.132	+1.217	+1.236

π_1 = 0.5 , π_1 = 0.5

Sigma :

7.96	5.39	4.72	5.19	4.23	5.79	5.87	5.72	4.7	5.34	5.52	5.04	5.15	6.26	5.34	5.78	5.53	5.65	5.53	5.96
5.39	7.2	4.92	5.31	4.95	6.03	6.47	5.48	4.62	5.01	4.74	5.02	4.86	6.1	5.5	5.9	5.3	5.14	6.02	5.71
4.72	4.92	6.97	5.38	4.88	4.9	6.08	4.64	4.63	4.88	4.82	5.11	5.25	6.13	5.33	5.01	5.61	4.54	5.8	5.87
5.19	5.31	5.38	6.84	4.25	5.68	6.34	5.51	4.85	5.49	5.03	5.01	5.11	5.63	5.32	5.52	5.45	5.35	5.86	5.9
4.23	4.95	4.88	4.25	5.8	4.73	5.02	4.47	3.63	3.99	3.69	4.16	4.24	4.83	5.18	4.83	4.36	4.54	5.22	4.98
5.79	6.03	4.9	5.68	4.73	7.87	6.7	5.43	5.54	5.55	4.8	4.96	5.36	6.5	5.58	5.93	5.72	5.56	6.49	6.21
5.87	6.47	6.08	6.34	5.02	6.7	8.46	6.03	5.8	6.08	5.87	5.88	5.64	6.98	5.88	6.32	6.	5.75	7.12	6.42
5.72	5.48	4.64	5.51	4.47	5.43	6.03	6.53	4.69	5.06	4.62	4.85	4.43	5.58	5.25	5.42	5.08	5.73	5.58	5.6
4.7	4.62	4.63	4.85	3.63	5.54	5.8	4.69	5.96	4.59	4.28	4.38	4.42	5.17	4.66	4.5	4.82	4.66	4.76	4.96
5.34	5.01	4.88	5.49	3.99	5.55	6.08	5.06	4.59	6.66	4.66	4.65	4.87	5.74	5.02	5.1	5.08	4.64	6.04	5.36
5.52	4.74	4.82	5.03	3.69	4.8	5.87	4.62	4.28	4.66	6.06	5.04	4.7	5.18	4.88	4.77	4.84	4.67	5.5	5.33
5.04	5.02	5.11	5.01	4.16	4.96	5.88	4.85	4.38	4.65	5.04	6.43	4.62	5.48	5.33	5.1	5.29	5.01	5.65	5.29
5.15	4.86	5.25	5.11	4.24	5.36	5.64	4.43	4.42	4.87	4.7	4.62	6.33	5.57	5.45	5.33	5.37	4.77	5.62	5.42
6.26	6.1	6.13	5.63	4.83	6.5	6.98	5.58	5.17	5.74	5.18	5.48	5.57	8.24	5.87	5.85	6.33	4.84	6.3	6.31
5.34	5.5	5.33	5.32	5.18	5.58	5.88	5.25	4.66	5.02	4.88	5.33	5.45	5.87	7.1	5.37	5.48	5.33	5.72	6.12
5.78	5.9	5.01	5.52	4.83	5.93	6.32	5.42	4.5	5.1	4.77	5.1	5.33	5.85	5.37	6.71	5.38	5.69	6.11	5.82
5.53	5.3	5.61	5.45	4.36	5.72	6.	5.08	4.82	5.08	4.84	5.29	5.37	6.33	5.48	5.38	6.97	4.85	5.73	6.25
5.65	5.14	4.54	5.35	4.54	5.56	5.75	5.73	4.66	4.64	4.67	5.01	4.77	4.84	5.33	5.69	4.85	6.96	5.54	5.61
5.53	6.02	5.8	5.86	5.22	6.49	7.12	5.58	4.76	6.04	5.5	5.65	5.62	6.3	5.72	6.11	5.73	5.54	8.19	6.33
5.96	5.71	5.87	5.9	4.98	6.21	6.42	5.6	4.96	5.36	5.33	5.29	5.42	6.31	6.12	5.82	6.25	5.61	6.33	7.89

5.2 Does k-NN classifier perform better than GDA or worse? Are there particular values of k which perform better? Why does this happen?

```
kNN is ready!
Randomly chosen result:
        k :  228     244     356     372     508     618     639     730     801
Accuracy : 52.38%  52.25%  52.62%  51.25%  51.50%  51.62%  52.50%  52.00%  50.75%
Best Accuracy in Validation Set is 57.00% w.r.t. 23 Neighbors.
kNN is ready!
```

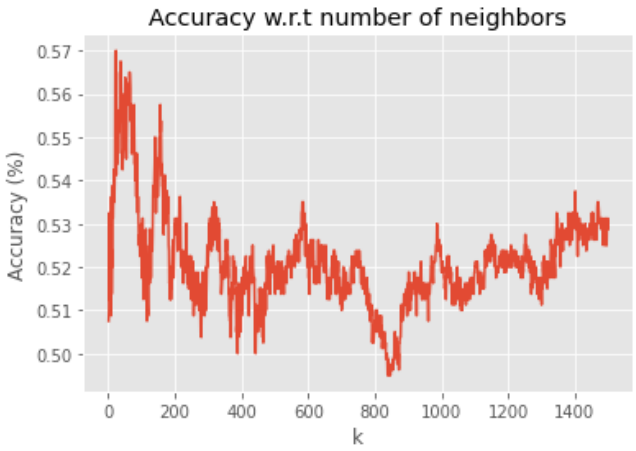

Figure 2

No. The result shows the accuracy of k-NN approximately not changed, but it still has no particular k as explained in 3.(a). In addition, the accuracy variation is more than the previous one as Figure 2 since the non-linearity of the model is more extensive. GDA results decreased drastically since the GDA considers one shared covariance matrix for all classes while DS2 comes from 3 different distributions for each class. In this case, the linear boundary failed for this non-linear problem.

5.3 Report the best fit accuracy achieved by this classifier.

```
Accuracy for Test Set is 51.00% w.r.t 23 Neighbors.
```

6 Comment on any similarities and differences between the performance of both classifiers on datasets DS1 and DS2?

In this assignment, we implemented GDA and k-NN as the linear and non-linear classifiers, respectively. The accuracy of GDA is more in the first part since the DS1 has a shared covariance matrix, and it matches with the assumption of GDA; however, DS2 is combinational of 3 different distributions, and it's harder to classify it with a linear boundary. Typically, GDA uses for non-linearity reduction. Therefore it is more efficient for identically distributed datasets.

Although the k-NN accuracy in DS1 is slightly greater than for DS2 because of less distribution, both are comparatively less than GDA. This can be because of the non-linearity assumption of k-NN, and increasing that has no significant effect on accuracy.

Part 2.

1.a Write down the equations for computing the mean and diagonal covariance matrices for the class conditional densities and the prior class probabilities using the maximum likelihood approach.

$$\ln P(D|\Pi, \mu_1, \mu_2, \Sigma_1, \Sigma_2) = \sum_{k\in K}\sum_{n\in k} \ln\left[\Pi_{\mathrm{k}}.N\left(x^{(n)}\middle|\mu_k, \Sigma_k\right)\right]$$

$$= \sum_{k\in K}\sum_{n\in k} \ln\left[\Pi_{\mathrm{k}}.\frac{1}{\sqrt{|\Sigma_k|}}e^{-\frac{1}{2}.\frac{(x^n-\mu_k)^2}{\Sigma_{\mathrm{k}}}}\right] + const. = \sum_{k\in K}\sum_{n\in k} \ln\left(\Pi_{\mathrm{k}}\right) - \frac{1}{2}\ln\left(|\Sigma_k|\right).-\frac{(x^n-\mu_k)^2}{2\Sigma_{\mathrm{k}}} + const.$$

Since the covariance matrix is diagonal: $|\Sigma_{\mathrm{k}}| = \prod_{i=1}^{n}\sigma_{i,k}^2 \rightarrow \ln(|\Sigma_{\mathrm{k}}|) = 2\sum_{j=1}^{M}\ln\left(\sigma_{j,k}\right)$.

$$\sum_{k\in K}\sum_{n\in k}\left[\ln(\Pi_{\mathrm{k}}) - \left(\sum_{j=1}^{M}\ln(\sigma_{j,k})\right) - \frac{1}{2}.\left(\sum_{j=1}^{M}\frac{\left(x_j^n-\mu_{j,k}\right)^2}{{\sigma_{j,k}}^2}\right)\right] + const. \quad (1)$$

$\boldsymbol{\sigma_{j,k}}$: Differentiate from eq. (1) w.r.t $\sigma_{j,k}$ and set it to 0:

$$\sum_{n\in k}[-\frac{1}{\sigma_{j,k}} + (\frac{\left(x^n-\mu_{j,k}\right)^2}{{\sigma_{jk}}^3}) = 0 \rightarrow \frac{-N_k}{\sigma_{j,k}} + \sum_{n\in k}\frac{\left(x_j^n-\mu_{j,k}\right)^2}{{\sigma_{jk}}^3} = 0 \rightarrow \sigma_{j,k} = \sqrt{\frac{\sum_{n\in k}\left(x_j^n-\mu_{j,k}\right)^2}{N_k}}$$

$\boldsymbol{\mu_{j,k}}$: Differentiate from eq. (1) w.r.t $\mu_{j,k}$ and set it to 0:

$$\sum_{n\in k}\frac{x_j^n-\mu_{j,k}}{{\sigma_{j,k}}^2} = 0 \rightarrow \sum_{n\in k}x_j^n = \sum_{n\in k}\mu_{j,k} \rightarrow \mu_{j,k} = \frac{\sum_{n\in k}x_j^n}{N_k}$$

$\boldsymbol{\Pi_{k'}}$: First I rewite the eq. (1) with releasing the $\sum_{k\in K}\Pi_k = 1$ constraint in direction of k for maximization w.r.t. $\Pi_{k'}$.

$$\sum_{n\in k}\left[\ln\left(1-\sum_{k'\in K-[k]}\Pi_{k'}\right)\right] + \sum_{k'\in K-[k]}\sum_{n\in k'}\ln[\Pi_{\mathrm{k}'}] + const.$$

Differentiate w.r.t $\Pi_{k'}$ and set it to 0:

$$= -\sum_{n\in k} \frac{1}{1-\sum_{k'\in K-[k]}\Pi_{k'}} + \sum_{n\in k'} \frac{1}{\Pi_{k'}}$$

$$\rightarrow -\frac{N_k}{1-\sum_{k'\in K-[k]}\Pi_{k'}} + \frac{N_{k'}}{\Pi_{k'}} = 0 \rightarrow$$

$$\frac{\Pi_{k'}}{\Pi_k} = \frac{N_{k'}}{N_k} \rightarrow \sum_{k'\in K} \frac{\Pi_{k'}}{\Pi_k} = \sum_{k'\in K} \frac{N_{k'}}{N_k} \rightarrow \frac{1}{\Pi_k} = \frac{N}{N_k} \rightarrow \Pi_k = \frac{N_k}{N}$$

1.b Estimate the parameters of the GNB model from the dataset. Report the best fit accuracy achieved.

The pixel in the corner is almost always black. Therefore, it is needed to add a σ_0 to the matrix to prevent zero variances. I decided to test different methods of adding a single value to all matrixes and setting a minimum value for the covariance matrix ($\Sigma_k = \max(\Sigma_k, \sigma_0)$). Figure 3, shows as the value of minimum increases, the accuracy of model is better. Its can be interpreted as the bigger number of minimum, the more deleted pixels.

For different values of σ_0 I calculated the accuracy in the Validation set and then chose the best σ_0. Next this value gets fixed, and the accuracy in the Test set is calculated. The second scenario I learned the model by replacing the zero variance values with 0.001 which considered more features.

First Scenario: Training the model with hyperparameter σ_o

The Accuracy on Validation set is: 84.22% w.r.t. σ_o=0.204

	1	2	3	4	5	6	7	8	9	10
π =	0.10	0.11	0.10	0.10	0.10	0.09	0.10	0.10	0.10	0.10
Number of Features =	333	164	346	319	301	331	297	275	313	273

The Accuracy on Test set is: 84.03%

Second Scenario: Setting the σ_o = 0.0010

The Accuracy on Validation set is: 63.16% w.r.t. σ_o=0.001

	1	2	3	4	5	6	7	8	9	10
π =	0.10	0.11	0.10	0.10	0.10	0.09	0.10	0.10	0.10	0.10
Number of Features =	557	554	600	568	576	576	553	570	550	540

The Accuracy on Test set is: 63.06%

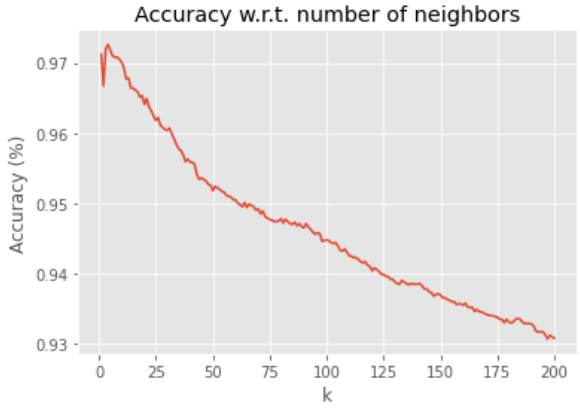

Figure 4

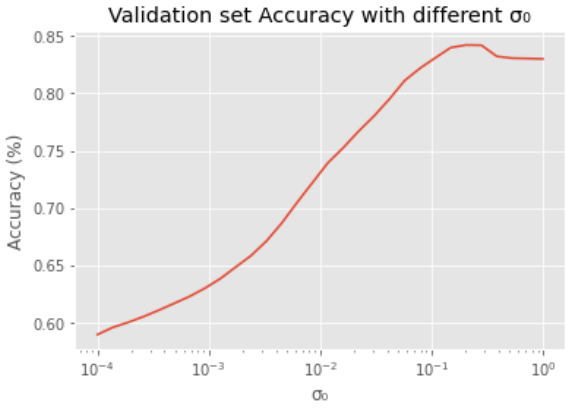

Figure 3

2 Use k-NN to learn a classifier. Repeat the experiment for different values of k and report the performance for each value.

```
• Progress: 100.00% , (10000/10000), Expected Remaining time: 00:00 •
Distances Calculated in 190.27 seconds.
kNN is ready!

Randomly chosen result:
       k :     3       6       11      20      21      53      56      60      64
Accuracy : 97.20%  97.10%  96.92%  96.41%  96.49%  95.20%  95.11%  95.05%  95.01%
       k :    76      79      99     110     112     125     143     163     186
Accuracy : 94.76%  94.75%  94.47%  94.25%  94.22%  93.99%  93.78%  93.52%  93.31%
```

Distance calculation for two sets of Training and Validation sets with 50000 and 10000 observations and 764 features takes a lot of time. Therefore, I implemented a parallelized CUDA with Numba library to calculate these values with GPU more efficiently. Numerous threads of GPU make it able to compute high volume basic mathematical calculations with more speed than CPU. With this implementation, it takes approximately 3 minutes on my laptop.

A single computation is = $x_i^{j|0} - x_i^{k|1} \quad i \in [1, \dots, M]\ (loop-3)$

$$j \in [1, \dots, N_{c0}]\ (sumloop)$$

$$k \in [1, \dots, N_{c1}]\ (loop-1)$$

Also, it is parallelized on sumloop, which has more speed than loop-1, and loop-3 is not parallelable with memory consideration. This code needs the Numba package and Nvidia CUDA Toolkit as well. To run in Colab, please use Q2_Colab file.

2.a Are there particular values of k which perform better? Why does this happen? Use validation accuracy for model selection.

```
Best Accuracy on Validation set is 97.26% w.r.t. 4 Neighbors.
```

Accuracy of the Validation set is the most in lower values of k, and it is maximum w.r.t. k=4. Why? The variance of clusters increases when they get bigger; therefore, accuracy gets off as the k is bigger (Figure 4).

2.b Report the best fit accuracy achieved by this classifier.

```
• Progress: 100.00% , (10000/10000), Expected Remaining time: 00:00 •
Distances Calculated in 175.89 seconds.
kNN is ready!
Accuracy for Test Set is 96.58% w.r.t. 4 Neighbors.
```

3 Compare the performance of GNB and k-NN for this MNIST classification. If one performs better over the other, explain why.

The accuracy is better in k-NN is better; however, it is not low for GNB. For both, the boundaries are non-linear, which is suitable for this problem. The GNB has a gaussian distributed assumption as well as independence between features. This can lead to some error in this method, while k-NN has a better performance without this assumption. However, more number of selected features in GNB results in less accuracy and it is approximately 63% for features with variances above 0.001.