Part 1 K=10

With Full Dimensions: After labelling according to most frequent digit in that cluster, with full Dimensions, I got overall Recall **60.943** and Precision **59.55** and after applying PCA so that residual variance is under 10%, I got slightly Increased overall Recall **61.021** and overall Precision **60.1.**From the below table Recall and Precision were **very high** for Digit **1** and **0;** But **very low** for digit 5,4,7,9. We can see from the table that there was confusion between digits **7&9 and 4&9 and 8&5 and 7&4**, that is also obivous as we can see that these digits are also very similiar in structure. This can also be verified from Output folder where Images are stored. This is also the reason why we got very high accuracy for 1 and 0.

After Applying PCA: After applying PCA I got **84 dimensions** with the condition that is mentioned above and accuracy increased slightly may be because other dimensions were actually either carrying very little information or more Noise. We can also observe that Precision and Recall actually increased for 5&9.

From table It can also be observed that those digits which were not similiar at all like 1&3 or 1&4... actually did not mix up while clustering.

Clusters with Full Dimensions

For 10:

Actual/Predict	0	1	2	3	4	5	6	7	8	9	Recall	60.943
0	152	0	4	1	0	2	0	1	1	0	94.41	
1	0	192	27	6	3	4	4	13	17	2	71.38	
2	0	6	127	6	0	0	2	2	3	0	85.81	
3	17	0	6	136	0	70	0	0	19	3	53.54	
4	2	1	5	1	87	2	10	19	4	41	49.43	
5	22	1	15	10	24	76	64	0	16	3	32.2	
6	5	0	9	2	2	1	120	0	0	0	82.76	
7	1	0	1	1	36	6	0	78	5	58	40.41	
8	0	0	5	31	0	28	0	0	132	2	64.08	
9	1	0	1	6	48	11	0	87	3	91	35.41	
Precision	76	96	63.5	68	43.5	38	60	39	66	45.5		
59.55												

Table 1

Clusters after applying PCA

For 10:

Actual/Predict	0	1	2	3	4	5	6	7	8	9	Recall	61.021
0	148	0	3	2	0	5	1	1	2	0	91.36	
1	0	192	27	3	3	3	4	13	18	2	72.18	
2	0	6	129	6	0	0	2	1	3	0	86.58	
3	10	0	7	138	0	66	0	0	18	4	56.1	
4	3	1	5	1	88	3	10	22	4	42	48.09	
5	28	1	13	10	21	78	55	0	14	2	34.36	
6	6	0	9	2	4	1	128	0	0	0	82.05	
7	1	0	1	1	36	5	0	77	6	56	40.53	
8	3	0	5	31	0	28	0	0	132	2	63.16	
9	1	0	1	6	48	11	0	86	3	92	35.8	
Precision	74	96	64.5	69	44	39	64	38.5	66	46		
60.1												

Table 2

Part 1 K=5

With Full Dimensions: K-means clustering with K=5 tried to mix up digits with similiar figures like **“4,7,9”, “2,6”, “3,8,5”, “0”, “1”**.

After applying PCA: Similiar results were obtained even after applying PCA, may be because K-Means has already put similiar digits in one group. We can also see that **Recall actually increased** in clustering of 5 because of the same reason as above.

In Main 2 Dimensions: Just to visualize clustering I took **2 Dimensions** in which variation was highest. It can be seen that how similiar digits has been mixed up in lower dimensions(Major Dimensions) and 0&1 are actually at opposite corner of graph. Graph clears about the results of K-Means clustering and similiarity between digits in feature space. **(Fig 1,2,3)**

Fig 4 is graph between cumulative Residual Variance and Features. It is clear that very few features can contain most of the variance in data hence can cluster better and faster in reduced dimensioned space.

Clusters with Full Dimensions

For 5:

Actual/Predict	0	1	2	3	4	5	6	7	8	9	Precisio
0	167	0	5	3	0	5	0	0	2	0	91.76
1	6	198	43	12	23	59	22	26	48	14	43.81
2	0	0	0	0	0	0	0	0	0	0	0
3	15	1	28	175	0	106	5	0	128	5	37.55
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	10	0	121	3	57	3	173	3	8	15	43.36
7	2	1	3	7	120	27	0	171	14	166	33.01
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
Recall	83.5	99	0	87.5	0	0	86.5	85.5	0	0	

Table 3

Clusters after applying PCA

For 5:

Actual/Predict	0	1	2	3	4	5	6	7	8	9	Precisio
0	165	0	5	3	0	5	0	0	2	0	91.67
1	6	198	47	15	23	60	25	27	52	14	42.31
2	0	0	0	0	0	0	0	0	0	0	0
3	16	1	24	172	0	106	5	0	124	5	37.72
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	11	0	121	3	53	2	170	3	7	15	43.48
7	2	1	3	7	124	27	0	170	15	166	32.57
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
Recall	82.5	99	0	86	0	0	85	85	0	0	

Table 4

Original Points in 2 Major Dimensions

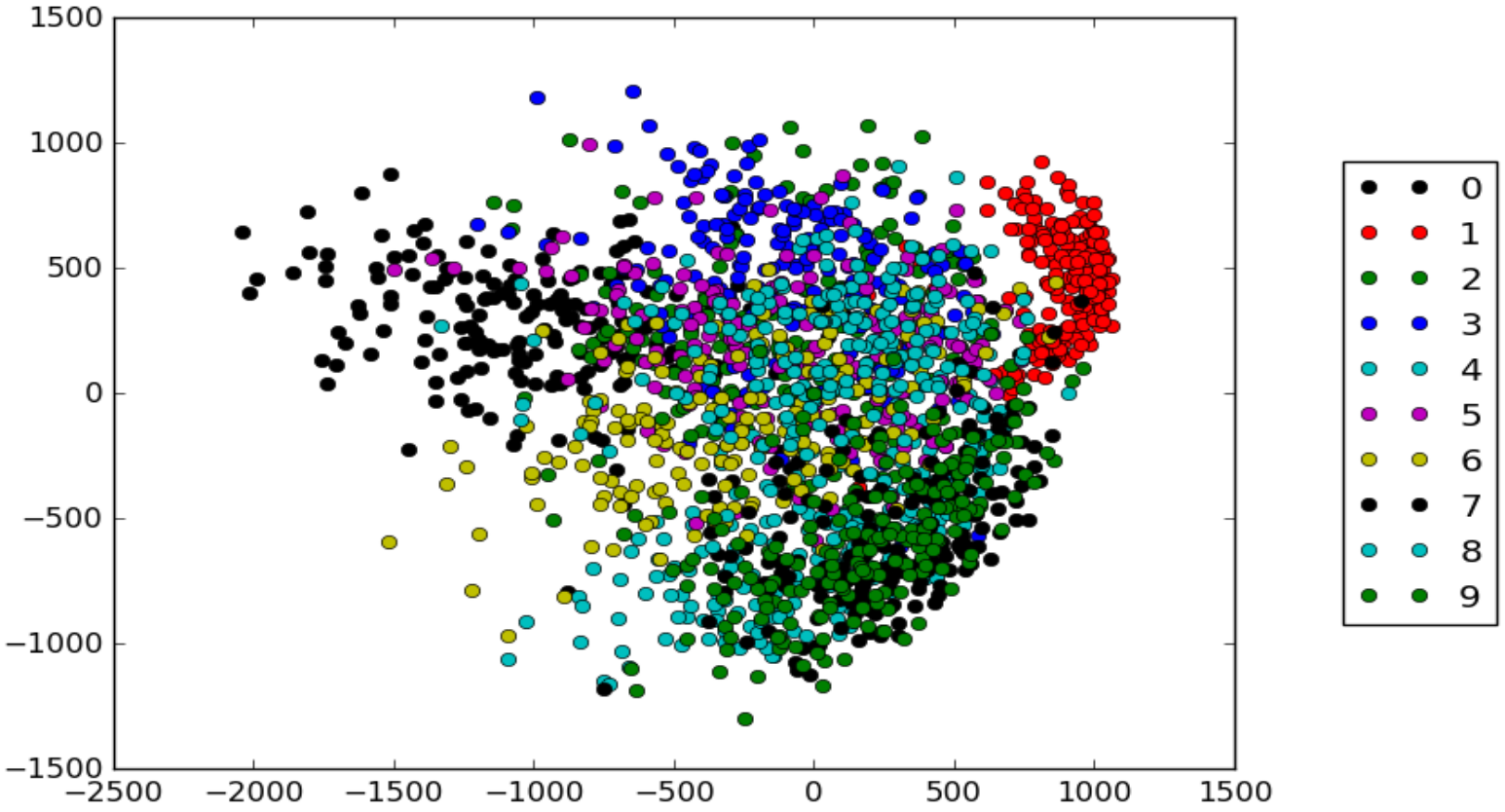

Fig 1

Predicted points in 2 Major Dimensions (Clustering in 10)

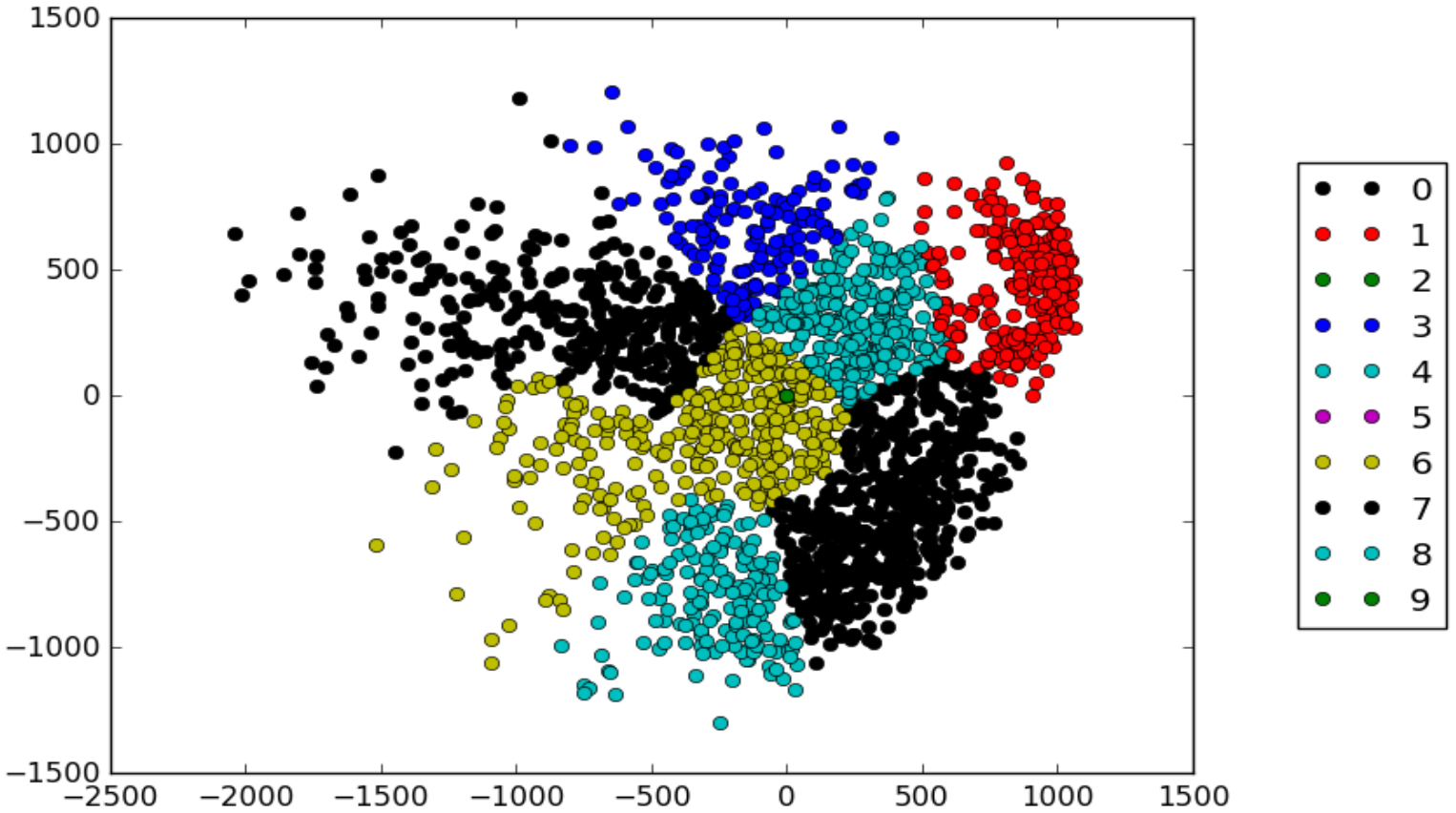

Fig 2

Predicted points in 2 Major Dimensions (Clustering in 5)

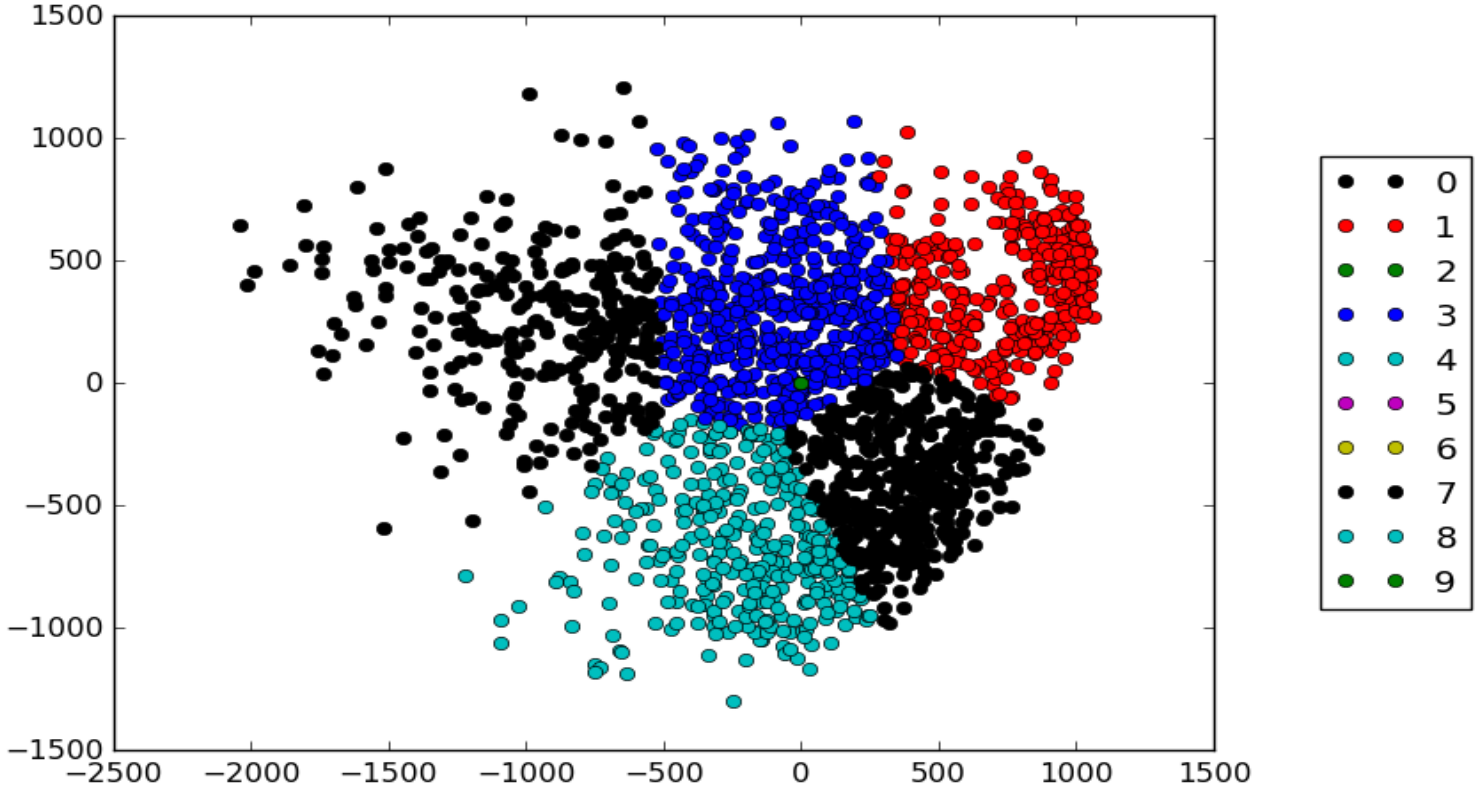

Fig 3

Residual Variance V/s Features

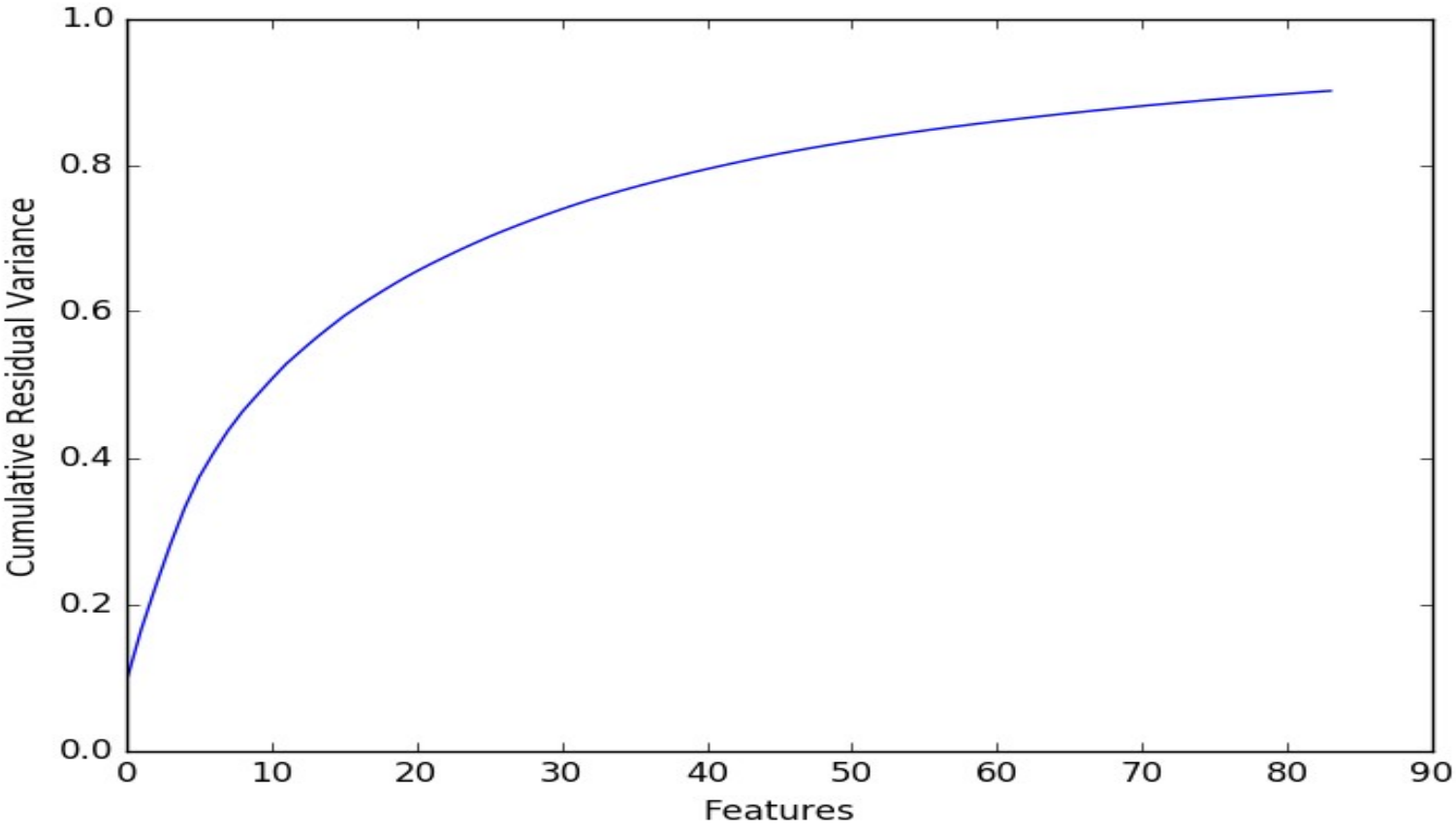

Fig 4

GMM

I tried to implement GMM for 2 clusters in 1-Dimensions. First I created random data from **random.gauss** in python with **mean,deviation = 0,2** and labelled it as **0** and **mean,deviation = 5,1** and labelled it as **1.** Then I started with choosing two random means and standard deviation. Then I calculated posterior probabilities of points in each clusters with starting equal **priors for each class** = **0.5.** After calculating posterior of each point I calculated priors of each class with given posteriors and normalised it and in this step I applied Expectation Maximisation. I observed that more and more iterations shifted mean towards actual means and standard deviations.

GMM actual increased accuracy over Kmeans because it is not hard clustering like Kmeans and actually cover property of features in terms of probabilities.

Formulas used in GMM can be easily seen in code.

Part 2

I tried various Clustering methods like **K-Means, GMM, Spectral Clustering, Affinity Propogation, Hierarcial Clustering** using **with PCA and without PCA**. Best solution that I got was with **17,18 Clusters,** with **GMM(1000 iterations, 10e-6 tolearance, 5 reinitialisation)** and with around **25 dimensions** after applying **PCA** which were covering around **.85 variation** in data. These settings were giving me accuracy around **0.67.** With Kmeans I was getting accuracy aroung **0.59** and could not increased more than that.

GMM actually **Increased** accuracy for this dataset over **Kmeans.**