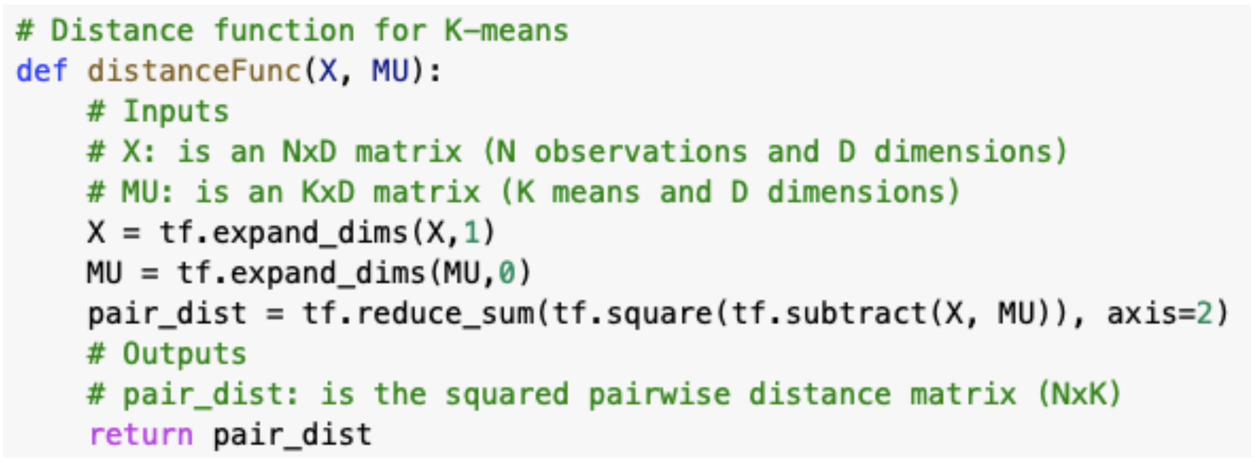
Assignment 3

1 K-means

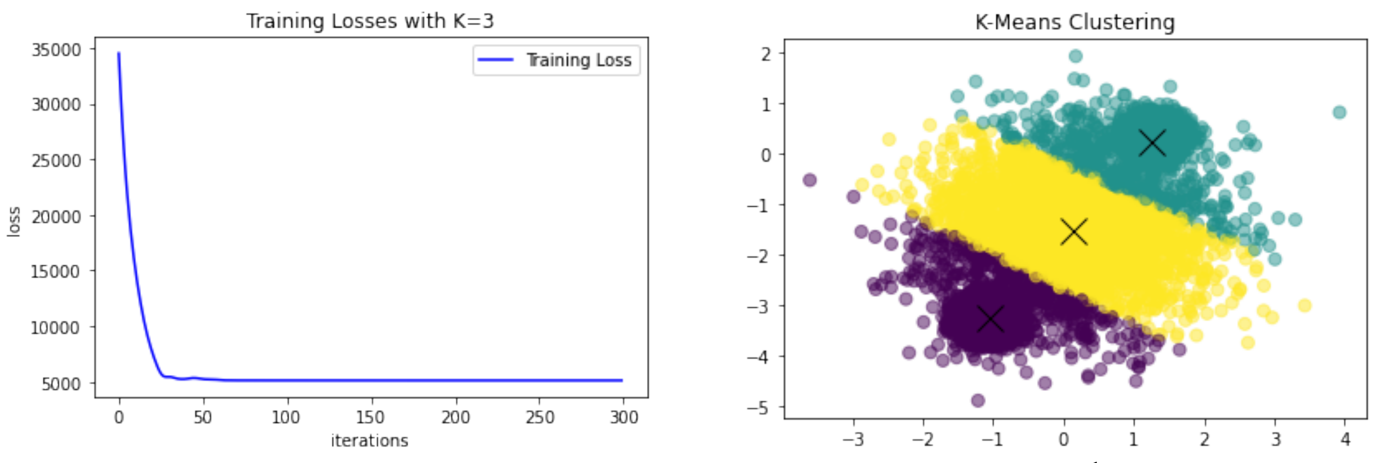
1.1 Learning K-means

1. In Figure 1 we see the distance function which helps us find the distances between the cluster centres and data points. After running the programming and training, we can see in figure 2a convergence after about 50 iterations and we can see the cluster organization in figure 2b. The cluster centres were sampled from a normal distribution with mean=0 and variance=1.



```
# Distance function for K-means
def distanceFunc(X, MU):
    # Inputs
    # X: is an NxD matrix (N observations and D dimensions)
    # MU: is an KxD matrix (K means and D dimensions)
    X = tf.expand_dims(X,1)
    MU = tf.expand_dims(MU,0)
    pair_dist = tf.reduce_sum(tf.square(tf.subtract(X, MU)), axis=2)
    # Outputs
    # pair_dist: is the squared pairwise distance matrix (NxK)
    return pair_dist
```

Figure 1. distanceFunc code in Python



a b

Figure 2 a) Plot of the loss with respect to number of iterations b) Plot of points divided into clusters

2. The plots below show the validation loss and clustering for the data with Ks 1-5 (Figure 3).

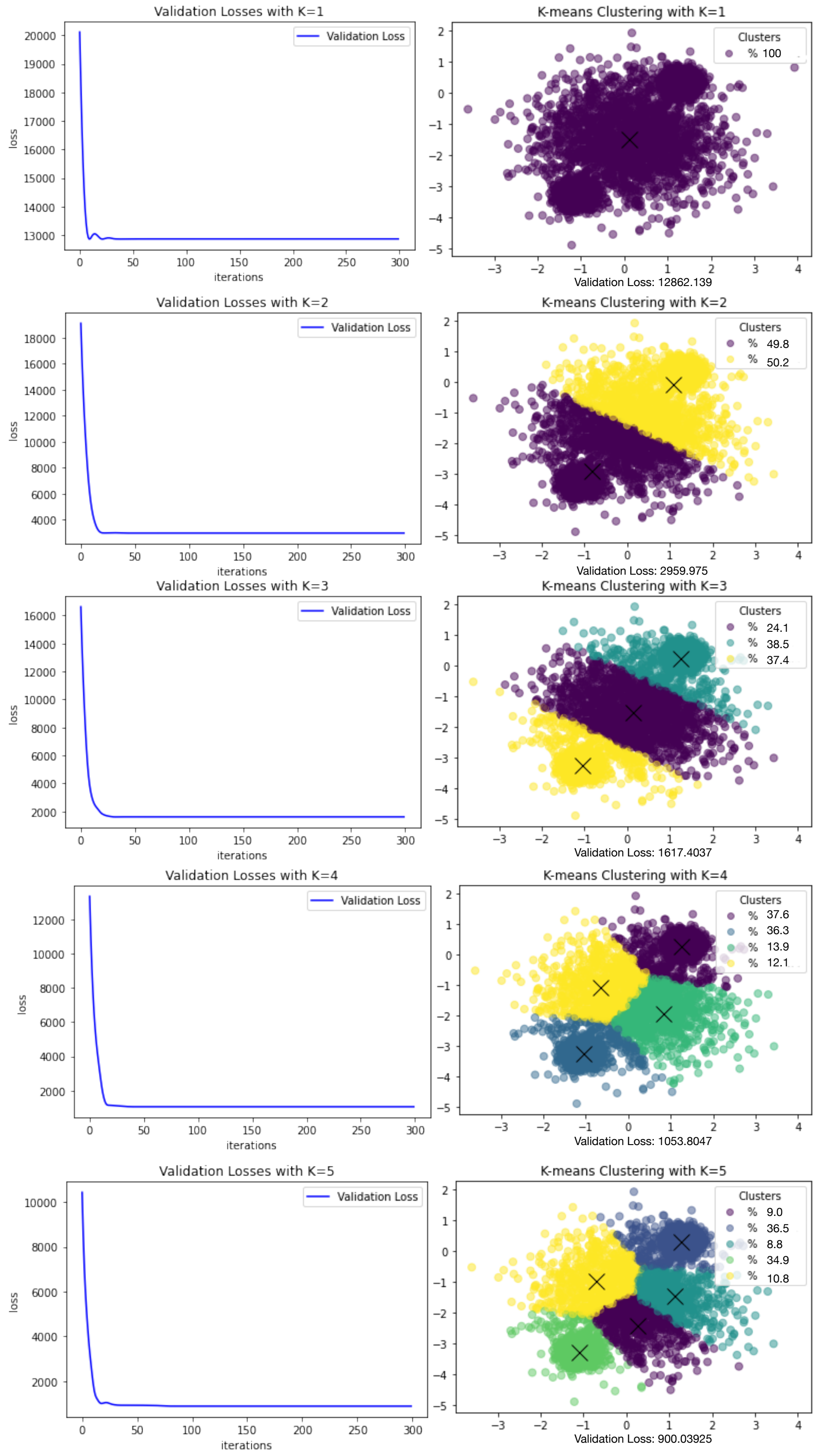


Figure 3. Validation Loss and Clusters plots for K=1-5

Table 1 shows what percentage of that data belongs to each cluster. As we can see in Table 2, the loss decreases as K increases. However, it's not correct to conclude that we should pick large K values to minimize the loss, even though if K is equal to data points then loss would be 0. Practically, that's not a sensible way to choose K, instead we look at how much the loss decreases with each K increase. After K=3, we don't see that much change in the loss (compared to how much it changed going from 1 to 2 and 2 to 3). Therefore, K=3 would be a good choice.

	K=1	**K=2**	**K=3**	**K=4**	**K=5**
Cluster 1	100%	49.7825%	24.1488%	37.6181%	9.0596%
Cluster 2	0%	50.2175%	38.4881%	36.2832%	36.5082%
Cluster 3	0%	0%	37.3631%	13.9493%	8.7896%
Cluster 4	0%	0%	0%	12.1494%	34.8733%
Cluster 5	0%	0%	0%	0%	10.7695%

Table 1. Percentage of that data belongs to each cluster for K=1-5

	K=1	**K=2**	**K=3**	**K=4**	**K=5**
Validation Loss	12862.139	2959.975	1617.4037	1053.8047	900.03925
ΔLoss		Δ_{21}=-9902.165	Δ_{32}=-2342.57	Δ_{43}=-563.6	Δ_{54}=-153.76

Table 2. Validation loss for each K value, and change in loss for each K increase

2 Mixtures of Gaussians

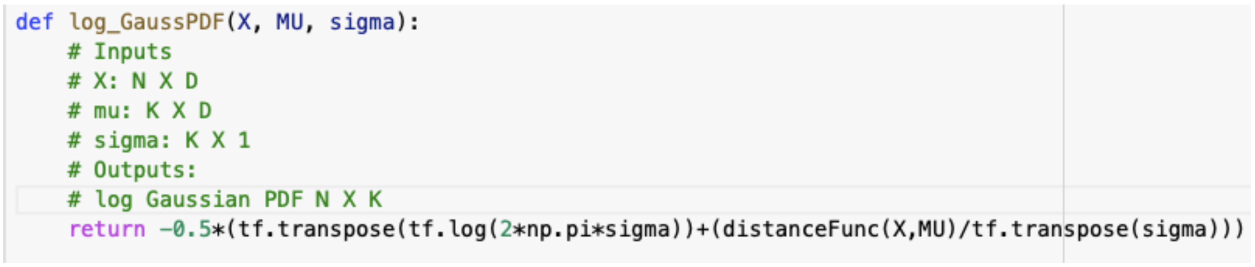
2.1 The Gaussian Cluster Mode

1. As we know the Gaussian distribution can be written as $g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-0.5(\frac{(x-\mu)^2}{\sigma^2})}$.

   Therefore:

   $log(g(x)) = log(\frac{1}{\sqrt{2\pi\sigma^2}}) - 0.5(\frac{(x-\mu)^2}{\sigma^2})) = -0.5log(2\pi\sigma^2) - 0.5(\frac{(x-\mu)^2}{\sigma^2}) = -0.5(log(2\pi\sigma^2) + (\frac{(x-\mu)^2}{\sigma^2}))$

   The code for the log_GaussPDF() is shown in Figure 4.



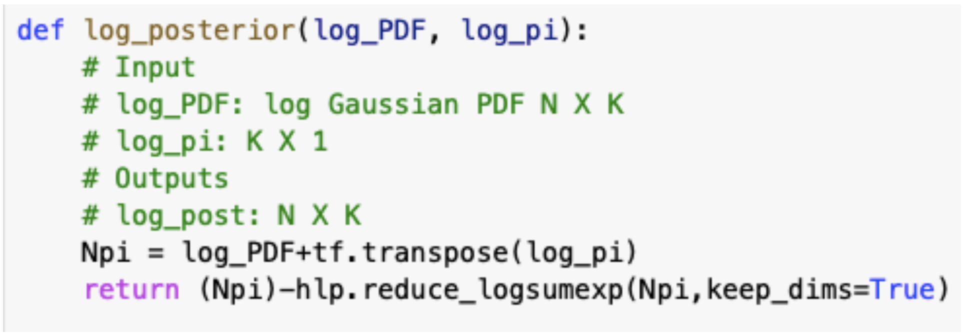
```
def log_GaussPDF(X, MU, sigma):
    # Inputs
    # X: N X D
    # mu: K X D
    # sigma: K X 1
    # Outputs:
    # log Gaussian PDF N X K
    return -0.5*(tf.transpose(tf.log(2*np.pi*sigma))+(distanceFunc(X,MU)/tf.transpose(sigma)))
```

Figure 4. Python code for the log_GaussPDF function

2. As we know $P(z\,|\,x) = \frac{N(x, \mu_k, \Sigma_k)\pi_k}{\Sigma N(x, \mu_k, \Sigma_k)\pi_k}$, therefore,

   $log(P(z\,|\,x)) = log(N(x, \mu_k, \Sigma_k)\pi_k) - log(\Sigma N(x, \mu_k, \Sigma_k)\pi_k)$

   The code for the log_posterior() is shown in Figure 5. We use the reduce_logsumexp() function instead of reduce_sum because we need the logarithm of a summation so basically log(reduce_sum()) which isn't the same as reduce_sum.



```
def log_posterior(log_PDF, log_pi):
    # Input
    # log_PDF: log Gaussian PDF N X K
    # log_pi: K X 1
    # Outputs
    # log_post: N X K
    Npi = log_PDF+tf.transpose(log_pi)
    return (Npi)-hlp.reduce_logsumexp(Npi,keep_dims=True)
```

Figure 5. Python code for the log_posterior function

2.2 Learning the MoG

1. As we know $P(X) = \Pi\Sigma\pi^k N(x, \mu^k, {\sigma^k}^2)$ and our loss function is $-log(P(x))$
The code for the loss is shown in Figure 6. We also initialize our sigma as tf.exp(tf.Variable()) to account for the constraint that $\sigma \in [0,\infty)$. Lastly, the cluster assignments are done using the argmax of P(z|x) (Figure 6).

```
clusters = tf.argmax(log_posterior(log_GaussPDF(X, MU, sigma), hlp.logsoftmax(pi)),1)
loss = -tf.reduce_sum(hlp.reduce_logsumexp(tf.transpose(hlp.logsoftmax(pi)) + log_GaussPDF(X, MU, sigma), 1, keep_dims=True))
```

Figure 6. Python code for loss and cluster assignment

We use the same Adam parameters as in K-means and sample from the same Gaussian as well (mean = 0, sd = 1). After training with K=3 we get the loss and clusters shown in Figure 7.

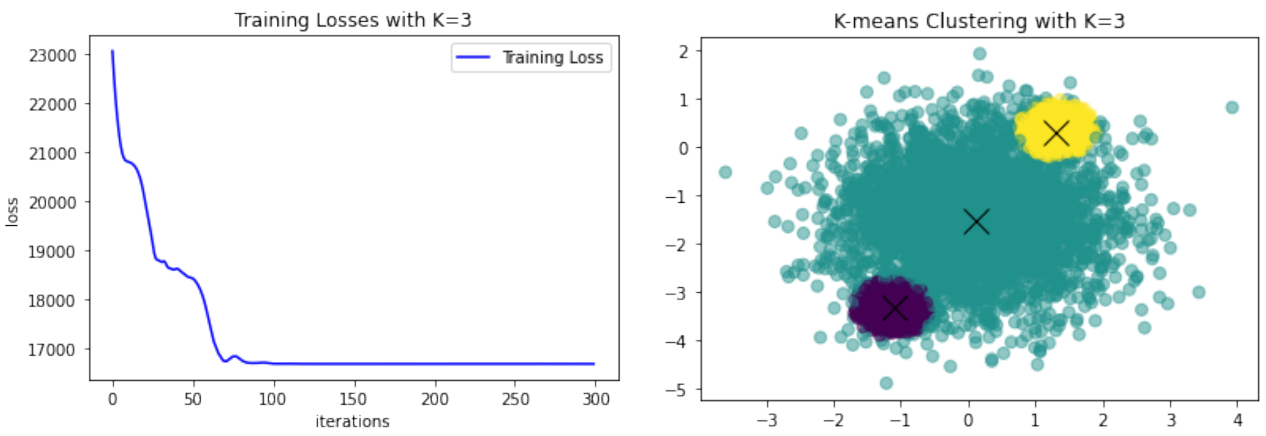


Figure 7. Training loss and clustering with K=3

The final optimized parameters are shown in Table 3.

Cluster	μ	σ	π
1	[0.101057, -1.52675]	2.573	-0.029
2	[-1.10363, -3.30806]	0.0740	-0.423
3	[1.29991, 0.311101]	0.741	-0.433

Table 3. Best model parameters for each cluster

2. The plots below show the validation loss and clustering for the data with Ks 1-5 (Figure 3).

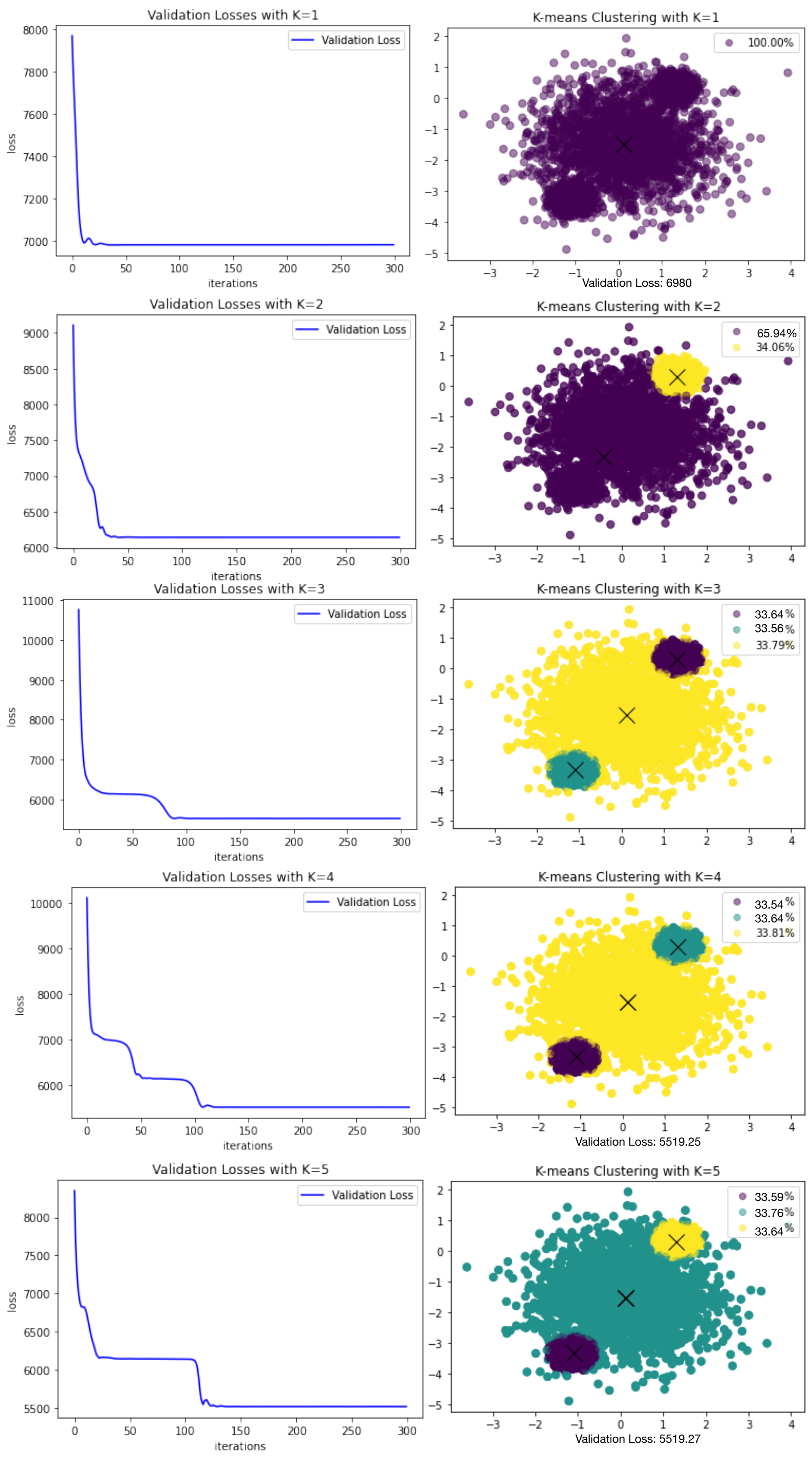


Figure 8. Validation Loss and Clusters plots for K=1-5

Table 4 shows what percentage of that data belongs to each cluster. Similar to the K-means results we shall choose K=3 as the best choice for our data, as we don't see that much change in the loss (compared to how much it changed going from 1 to 2 and 2 to 3) after K=3.

	K=1	**K=2**	**K=3**	**K=4**	**K=5**
Cluster 1	100%	65.94%	33.64%	33.55%	33.59%
Cluster 2	0%	34.06%	33.56%	33.64%	33.76%
Cluster 3	0%	0%	33.79%	33.8%	33.64%
Cluster 4	0%	0%	0%	0%	0%
Cluster 5	0%	0%	0%	0%	0%

Table 4. Percentage of that data belongs to each cluster for K=1-5

	K=1	**K=2**	**K=3**	**K=4**	**K=5**
Validation Loss	6980	6142.99	5519.39	5519.25	5519.27
ΔLoss		Δ_{21}=-837.01	Δ_{32}=-623.6	Δ_{43}=-0.14	Δ_{54}=-0.02

Table 5. Validation loss for each K value, and change in loss for each K increase

3. The overall loss is much lower in the GMM model compared to K-means. In both models, we see a drastic drop between K=5 and K=10, which means that the number of clusters across the data100D.npy is between 5 and 10. If we were too choose from this set of Ks, K=10 would be a good choice. Table 6 and Figure 9 report and depict the validation loss across all Ks for both models. You may notice that only 50 iterations were used, this is because both models converged after 50 and resulted in very similar final loss values which would make it hard to compare the K values.

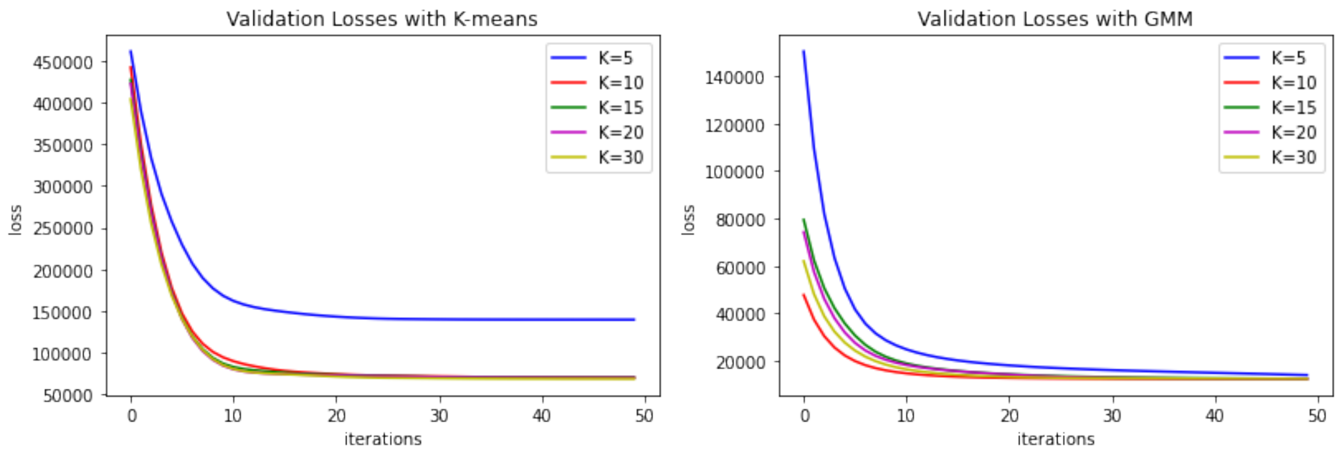


Figure 9. Validation Loss and Clusters plots for K=1-5

	K = 5	**K = 10**	**K = 15**	**K = 20**	**K = 30**
K-means	139351.98	69906.42	69928.18	69488.08	68316.2
GMM	14037.226	12432.257	12467.259	12431.774	12469.215

Table 6. Validation losses for both GMM and K-means across different K values