1. When training the probabilistic generative classifier, how does the full covariance compare to diagonal covariance in performance for each of the data sets? Why?
With respects to the HS dataset, they gave really low results which could be facts such as hardcoding of priors from a multiclass dataset such as HS is not an easy task in any way. Regarding the 2D dataset, the difference is really small and for 7D is actually no difference. I tried different sets of random seeds to pick different ranges for it, but the result was not really different.
I decided to train my model using the diagonal covariance generative probabilistic method to predict the test labels. I think diagonal covariance is making the features independent which could help in the classification task. According to Ng and Jordan, when we are using the full covariance matrix we are more on danger of singular matrix if we have fewer than linear among our n training examples. Therefore, I decided to stay with the diagonal as they are getting similar results.

Full Covariance one -For 2D

Fold	1	2	3	4
accuracy	0.972	0.976	0.98	0.984
average	0.978			

Diagonal Covariance one -For 2D

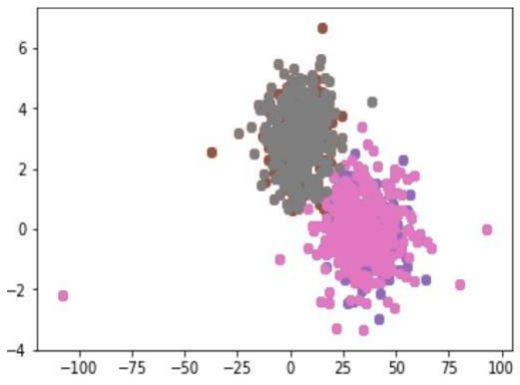
Fold	1	2	3	4
accuracy	0.972	0.976	0.984	0.976
average	0.977			

Full Covariance one -For 7D

Fold	1	2	3	4
accuracy	1.00	1.00	1.00	1.00
average	1.00			

Diagonal Covariance one -For 7D

Fold	1	2	3	4
accuracy	1.00	1.00	1.00	1.00
average	1.00			



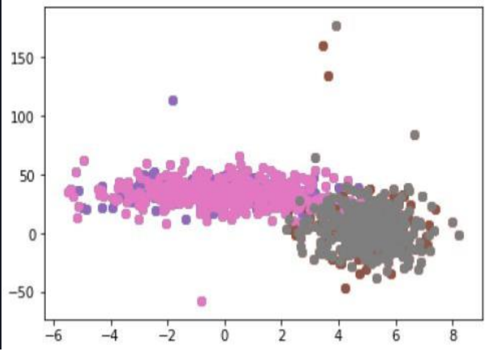


Figure1_a is the distribution plot for the 2D dataset using the Diagonal Generative Probabilistic model (with 4-fold cross validation), Figure1_b is the distribution plot for the 2D dataset using the Diagonal Generative Probabilistic model (with 4-fold cross validation)

2. When training KNN classifier, what happens as you vary k from small to large? Why?

I used 4-fold cross validation to test the accuracy of my model according to different variations of training and validation set. For each dataset, I am computing the accuracy mean across different folds with respect to different values of k.
According to [9], if the number of classes in a classification task is two, it would be better to training the K-Nearest Neighbor (KNN) using odd values of k. This could cause a problem for the cases when you reach the same number of labels for each class. Therefore, I trained the 2D and 7D datasets using the odd numbers in rage of [3,20) while I am training Hyperspectral dataset using both all number in range of [3,20) to investigate if it would affect the final prediction accuracy or not.
As it is shown in the following tables, for the binary classification (only two unique classes) we are able to reach the highest accuracy using small numbers of k(k=3). However, when the number of different classes increased, in the Hyperspectral case, we need more data to be able to train our models well enough.
I found many different discussions through the web about the proper method to get the optimum "k" for KNN. However, I think it would be a good step to use cross validations to train different splits of your model with different values of the k. This would help anyone to see that what amount of "k" is cancelling the noise and achieves better results. Therefore, I am using k-fold cross validation and keep track of different accuracy mean which help me in deciding what is the best one.
You could see that increasing "k" from small to large could have different outcomes in regard to different datasets. Once you go with small values of k, the noise will have a higher influence on the classification result. On the other side, large values are computationally expensive as KNN is a non-parametric and instance-based approach. Let's look at "k" as a tool which helps us in focusing on a part of the sample data points to decide what is the label of the unknown input (Test data). This picture would help us to find out what would happen if the k is too small or too large. If k is too small (such as k=1), we are not obtaining really that much information from the whole distribution which could be helpful in getting better results. On the other hand, if k is too large, we are including more neighbors in our labeling process which they could be actually outliers. The large values of k introduce biases intro the problem which may affect the classification results significantly. As my experiment results also validated, by increasing "k" from small to large the results are getting better most of the times until we reach the optimum k in that range. I used numbers between [3,20) as I had different datasets and I intended to test them upon the same set of metrics and constraint. It would be possible to make the range less, but I thought that could may affect the results and validity of this experiment.
I decided to use KNN for Hyperspectral dataset as it was giving best results. I believe that this could be due to the fact that generating the best prior for a complex dataset like HS one could not be easy. As it is suggested in [10], the KNN is easier to use for a multiclass like dataset such

as HS one.
2D Dataset:

K	3	5	7	9	11	13	15	17	19
Mean accuracy	0.971	0.97	0.97	0.969	0.967	0.964	0.964	0.961	0.96

7D Dataset:

K	3	5	7	9	11	13	15	17	19
Mean accuracy	0.995	0.994	0.994	0.992	0.991	0.99	0.99	0.988	0.986

HS Dataset (with odd values of K for KNN model) :

K	3	5	7	9	11	13	15	17	19
Mean accuracy	0.813	0.814	0.828	0.825	0.827	0.828	0.834	0.833	0.833

HS Dataset (with both odd and even values of K for KNN model):

K	3	5	7	9	11	13	15	17	19
Mean accuracy	0.814	0.823	0.83	0.833	0.823	0.83	0.839	0.832	0.84

References:

3.https://stackoverflow.com/questions/2600191/how-to-count-the-occurrences-of-a-list-item
4. https://stackoverflow.com/questions/268272/getting-key-with-maximum-value-in-dictionary
5.https://www.wikiwand.com/en/Generative_model
6. http://scikit-learn.org/stable/modules/cross_validation.html
7. https://docs.scipy.org/doc/numpy/reference/generated/numpy.diag.html
8. http://scikitlearn.org/stable/modules/generated/sklearn.model_selection.KFold.html
9. https://discuss.analyticsvidhya.com/t/why-to-use-odd-value-of-k-in-knn-algorithm/2704 10. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/
11. https://www.wikiwand.com/en/Generative_model