CS5691: Pattern Recognition and Machine Learning
Programming Assignment II

PATTERN CLASSIFICATION

September 23, 2022

Aanand Krishnan *(BE17B001)*
Manoranjan J *(NA17B112)*
Reneeth Krishna MG *(BS17B025)*
Team 17 - CCS Section Spring 2021

Contents

List of Tables

List of Figures

Classification

In the previous assignment we performed regression, in which the target variable was simply a real valued vector. However, in classification the task is of organizing data into categories for it's efficient use. A shorthand way of saying this would be, the output variable in regression takes continuous values, whereas in classification the output variables takes class labels. For the classification problem at hand, the input feature vector to the model takes numerical attributes. The task of the model then is to assign a class label for the given feature vector.

To build an intuition for the process of classification one could think of a numerical input that has only two features. *For example*, height and weight of several individuals as input for classification as adult and child. Now the classification task is to assign a class (child or adult) to a new unlabelled observation. Such an 2D features input can be easily visualized and the class of a new observation can be inferred from the plots by a very reasonable assumption that the class of the unlabelled point is determined predominantly by nearby points of the training dataset. An interesting observation to note is that, there was no intrinsic geometry or vectors in the example of classification mentioned above or for any classification task for that matter, but by viewing them as vectors we get a geometric intuition that is extremely helpful.

However a severe pitfall of this approach is outlined in chapter one of Bishop. The author points out how with an increase in the number of features, the nearby distance we would scan to make an classification also increases drastically. Such an increase in distance demands an equally drastic increase in number of training data available, making the process infeasible for more than a few featured input. This difficulties arising in higher dimensions is referred to as *curse of dimensionality* in literature.

This curse however doesn't impair our ability to build machine learning models for real world data that has several thousands features. Bishop, points out a reason for this is that we could essentially capture the variance shown by the data with a subset of features projected on to a different space which has lower dimensions, thus confining the data. Now that we are aware of the effect of dimension, a brief account of how we could model a classification task is provided below.

Bayesian Decision Theory

One could abstract the entirety of the models used in machine learning as being developed from minimising different kind of loss functions. In function approximation we minimised the squared distance loss function to make predictions. Bayesian Decision Theory for classification task, is based on a probabilistic loss function. In other words the problem is posed in Probabilistic terms.

A supervised classification task involves labelled training data,

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

. The training data is used for learning, following which one of the following methods is followed for prediction,

- Using Bayes Theorem,

  $$p(Y = y_i|\bar{x}) = \frac{p(\bar{x}|y_i).p(y_i)}{p(x)}$$

  Now we estimate the posterior class probabilities $p(x|y)$. We can use the joint distribution $p(x, y)$. This kind of methods in which attention is paid to probability distributions of both x and y are known as *generative models.*

- In the previous case we were given an x, which needed classification. What if we could find a function f, that could map each inputs of x to a class label? Such a function, f is called *discriminant.* This kind of models are shown to computationally efficient than the former. A discriminant function can be any one of the following,

  $$g_i(\bar{x}) = \begin{cases} p(Y = y_i|\bar{x}), \\ p(\bar{x}|Y = y_i).p(Y = y_i), \\ \ln p(\bar{x}|Y = y_i) + \ln p(Y = y_i) \end{cases}$$

  Where the class label i is chosen for which $g_i(\bar{x})$ is maximum. The discriminant function can also be used to find the decision region for a class. $g_i(\bar{x}) - g_j(\bar{x}) = 0$, gives the decision surface seperating the two classes i and j.

Gaussian Mixture Model

This is a discriminant model. In determining the discriminant function several assumptions need to be made, one major assumption of this model is that,

$$p(\bar{x}|y_i) = \sum_{q=1}^{Q} w_q \mathcal{N}(\bar{x}|\bar{\mu}_{iq}, \bar{c}_{iq})$$

That is, the class likelihood function is assumed to be a multimodal multivariate Gaussian distribution with $Q-$ peaks. Such an distribution better suits the real world data, for that too contains multiple peaks opposed to the unimodal Gaussian which has a single peak. w_q is scaling factor to adjust for the peak of the distribution with the peak observed in data.

$$\text{Number of parameters for class i} = Q_i \left(1 + d + \frac{d(d+1)}{2}\right)$$

We could further make assumptions to reduce the number of parameters such that the covariance matrices are equal or covariance matrices are diagonal (*naive* Bayes classifier) and such. Posterior probability for a point to belong a Gaussian component of a particular class is denoted by γ_{nq} and is also called the responsibility term. An expression for the responsibility term is,

$$\gamma_{nq} = \frac{w_q \mathcal{N}(\bar{x}|\bar{\mu}_q, \bar{c}_q)}{\sum_{m=1}^{Q} w_m \mathcal{N}(\bar{x}|\bar{\mu}_m, c_m)}$$

Maximising this responsibility term using gradient descent we will be able to find out the various parameters of this model. An another advantage of GMM is there might be cases where two classes that are not linearly seperable have the same covariance matrix resulting in a linear decision boundary while using a single Gaussian thus resulting in a poor classification. However there is a hiccup in the parameter estimations for GMM, as it turns out the parameters of this model don't have closed form expression and hence an iterative method known as Expectation-Maximisation(EM) is used.

Steps involved in EM method

Initialization

- Instead of choosing random values, we divide the data into disjoint sets(clusters) using suitable algorithms such as K-Means. The number of clusters K will be the the hyperparameter Q. The rest of the parameters can be estimated as follows,

$$\begin{aligned} N_q &= \text{No. of examples in the } q^{th} \text{ cluster} \\ w_q &= \frac{N_q}{N} \\ \gamma_{nq} &= \begin{cases} 1, & \text{if } \bar{x}_n \in \text{q}^{th} \text{ cluster,} \\ 0, & \text{otherwise} \end{cases} \\ \bar{\mu}_q &= \frac{1}{N_q} \sum_{n=1}^{N} \gamma_{nq}.\bar{x}_n \\ c_q &= \frac{1}{N_q} \sum_{n=1}^{N} \gamma_{nq}(\bar{x}_n - \bar{\mu}_q).(\bar{x}_n - \bar{\mu}_q)^t \end{aligned}$$

- The set of initial parameters $\bar{\theta}^{old}$ is passed on to the next step.

$$\bar{\theta}^{old} = \{w_q, \bar{\mu}_q, c_q\}_{q=1}^{Q}$$

Expectation

- Update the responsibility term as follows,

$$\gamma_{nq} = \frac{w_q \mathcal{N}(\bar{x}_n|\bar{\mu}_q, c_q)}{\sum_{m=1}^{Q} w_m \mathcal{N}(\bar{x}_n|\bar{\mu}_m,}$$

Maximization

- Using the responsibility term obtained in the expectation step, we can estimate N_q which can further be used in the expressions for the other parameters thus updating them all.

$$N_q = \sum_{n=1}^{N} \gamma_{nq}$$

- The new set of parameters are send on to the next step. The convergence can't be looked at because of the large number of parameters involved. Instead we use a threshold for the stopping criterion as explained in the next step.

$$\bar{\theta}^{new} = \left\{ w_q^{new}, \bar{\mu}_q^{new}, c_q^{new} \right\}_{q=1}^{Q}$$

Likelihood

- We estimate the log likelihood as follows,

$$\begin{aligned} \mathcal{L}(D) &= \sum_{n=1}^{N} \ln p(\bar{x}_n | \bar{\theta}) \\ &= \sum_{n=1}^{N} \ln \sum_{q=1}^{Q} \mathcal{N}(\bar{x}_n | \bar{\mu}_q^{new}, c_q^{new}) \end{aligned}$$

- We will look at the increase in likelihood for each updated set of parameters and stop when the difference is below a certain threshold. Else the previous steps are repeated. A guarantee of the EM method is that the the change in the log likelihood will always be positive.

Note: The parameters estimated using EM method are not unique, as what we obtain is a local maximum of the likelihood and not necessarily a global maximum. Hence, the value of the parameters estimated using this method depends on our initialization. Different values of $\bar{\theta}^{old}$ gives different estimates.

Naive Bayes Classifier

This is a special case of the discriminative methods described above. In this model, the density function assumed is a multivariate gaussian function. There is one more important assumption which states that the features are conditionally stastically independent which means that the covariance matrix of the gaussian function is always diagonal which results in lesser parameters to estimate. This function may work well for unimodal data and data in which the features are independent. If the features x_k of the input vector, $\bar{x} = (x_1, x_2, \ldots, x_d)$ are independant then,

$$\begin{aligned} p(\bar{x}|y_i) &= p(x_1, x_2, \ldots, x_d|y_i) \\ &= \prod_{k=1}^{d} p(x_k|y_i) \\ &= \prod_{k=1}^{d} \mathcal{N}(x_k|\mu_{ik}, c_{ik}) \end{aligned}$$

Mean and covariance are the parameters to be estimated for each class. Since the number of parameters may become high if we consider unique covariance matrices, we sometimes assume the covariance matrix to be same for all classes and in some cases we even assume the diagonal terms are same for the covariance matrices involved to reduce the number of parameters to estimate. These assumptions has its own limitations during the prediction though.

K Nearest Neighbours for density Estimation

In this method we assume a different density function than the GMM/naive bayes models we looked at in the previous section.

$$p(\bar{x}|y_i) = \frac{K}{N.V}$$

Here K is the nearest neighbours to the point in consideration($\bar{x}$) and is fixed for a particular model. N is the total number of examples for the class. V is the volume that is to be determined which encloses all the K neighbours. Given a particular class and the value K, the only varying entity is the volume V for different data points and this plays a role in determining the class for that particular point. This model comes under the non-parametric methods for density estimation which means that there is no weights which has to be optimised for the given data and the prediction of class is based on linear algebra and probability. Thus the class label of $\bar{x}$ upon simplification of the bayes rule is determined by:

$$p(y|\bar{x}) = argmin(V_i)$$

KNN classifier

This model also comes under non-parametric method for density estimation. We will be assuming the same expression for $p(\bar{x}|y_i)$. But now we try to identify K nearest neighbours for a particular point in consideration based on all the examples and check the number of the neighbour points to different classes and assign the class which has the largest neighbours out of the K points. Thus the class label of $\bar{x}$ upon simplification of the bayes rule is determined by:

$$p(y|\bar{x}) = argmax(K_i)$$

I. Pattern Classification on Linearly Separable Data

I.1. K-Nearest Neighbours Classifier

The given synthetic data is classified using the KNN Method. The data is already well separated into classes. Thus a very good performance for even K=1 is observed. A 100% accuracy is observed in all training, validation and test set for all the tested values of K. The decision region is also plotted and the training data is overlapped with it. Let us look at some interesting plots obtained for the KNN method for the given dataset.

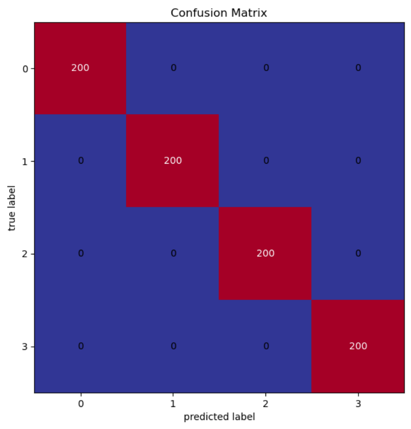
The decision plot for K=1 is shown in [2]. As can be observed, the decision boundary is slightly distorted for lower values of K. On the other hand, as the values of K increases we get a more smoother decision boundary as can be seen in [6] for K=15.

Model	Training Accuracy	Val Accuracy
K=1	100%	100%
K=7	100%	100%
K=15	100%	100%

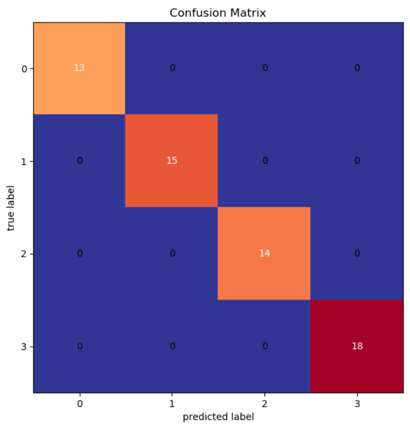
Table 1: Performance of various KNN Models

.

Plots for K=1



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 1: Confusion Matrix for KNN Model trained with K=7

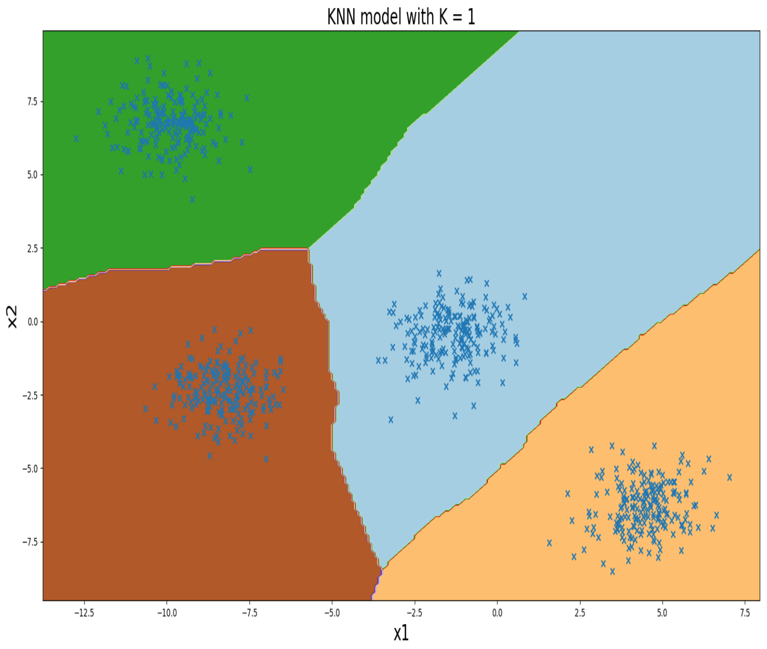
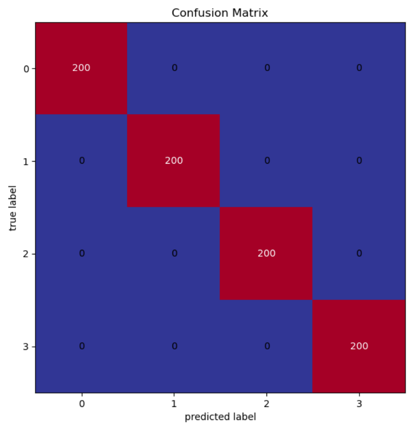
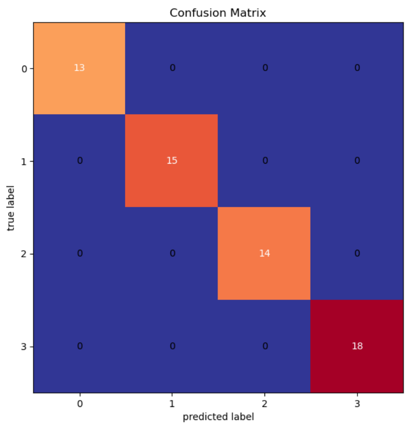


Figure 2: Decision Plot KNN Model trained with K=1

I.1.1. Plots for K=7



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 3: Confusion Matrix for KNN Model trained with K=7

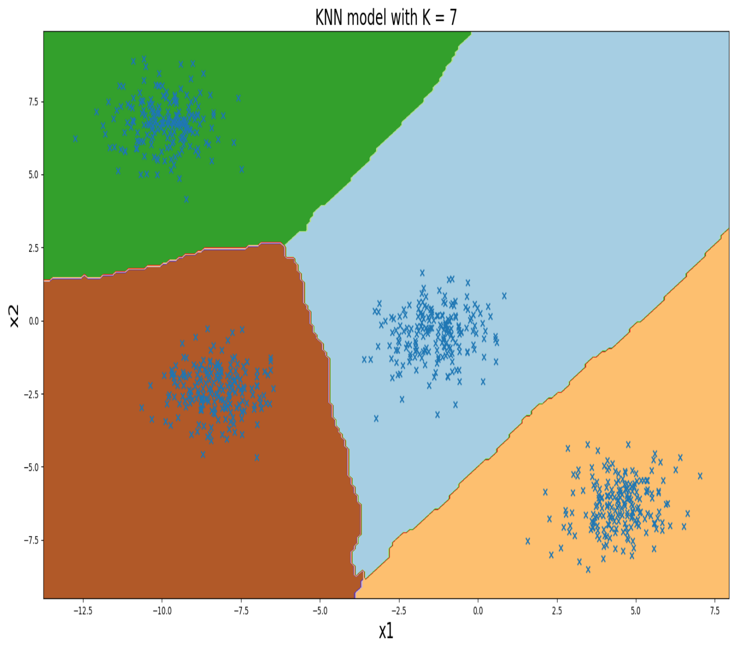
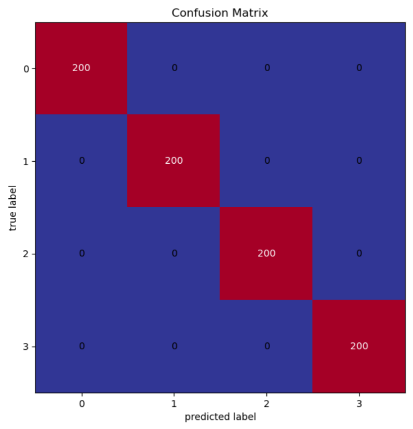
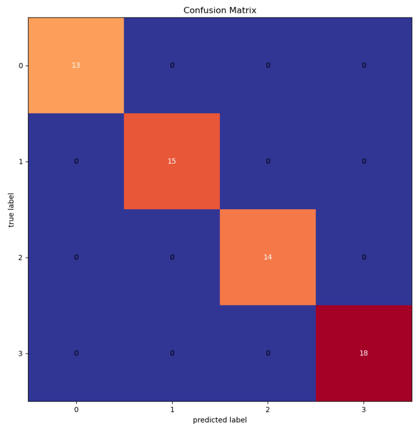


Figure 4: Decision Plot KNN Model trained with K=7

I.1.2. Plots for K=15



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 5: Confusion Matrix for KNN Model trained with K=15

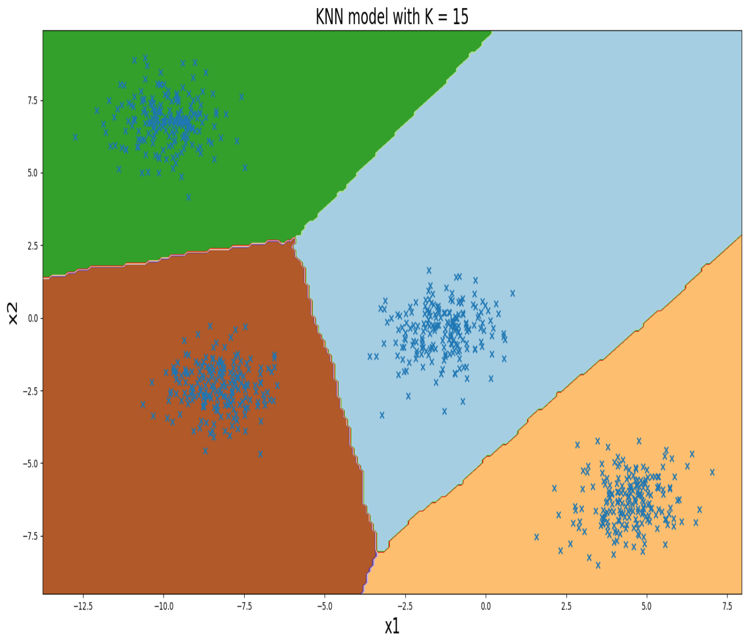


Figure 6: Decision Plot KNN Model trained with K=15

I.2. Naive Bayes Classifier with Gaussian Distribution

The covariance matrix is a diagonal matrix in the case of naive bayes classifier and the nature of the matrix is experimented in this section to study about the various observations in the decision plot. Let us look at several interesting plots.

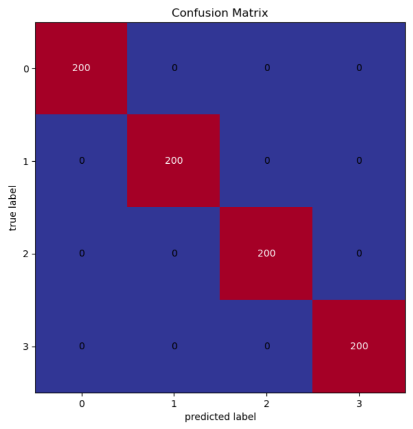
Similar to the case of KNN, the model accuracies didn't vary much in this case well since it is a synthetic data and we get very good results as can be observed from the confusion matrices and [2]. The decision boundary in all the 3 cases are observed to be linear since we assume that the covariance matrix is diagonal. The level curves of the gaussian function assumed is a circle/ellipse with its axes parallel to the coordinate axes in all the 3 cases. Case 1: [8] has a circular level curve. All other cases have an elliptical level curve. Since the variance is very small, the elliptical nature is not directly visible from their plots. Here *C1, C2* represent the covariance matrix for different classes. For the sake of convenience we only represent the conditions using 2 classes but in our case, we have 4 classes involved.

Model	Train Accuracy	Val Accuracy
$C1 = C2 = \sigma^2 I$	100%	100%
$C1 = C2$	100%	100%
$C1 \neq C2$	100%	100%

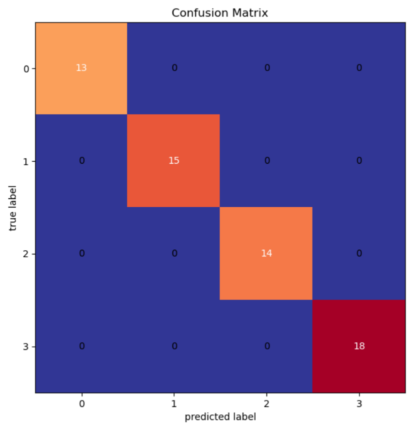
Table 2: Performance of various Naive Bayes Models

.

I.2.1. Plots for $C1 = C2 = \sigma^2 I$



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 7: Confusion Matrix for Naive Bayes Model trained with $C1 = C2 = \sigma^2 I$

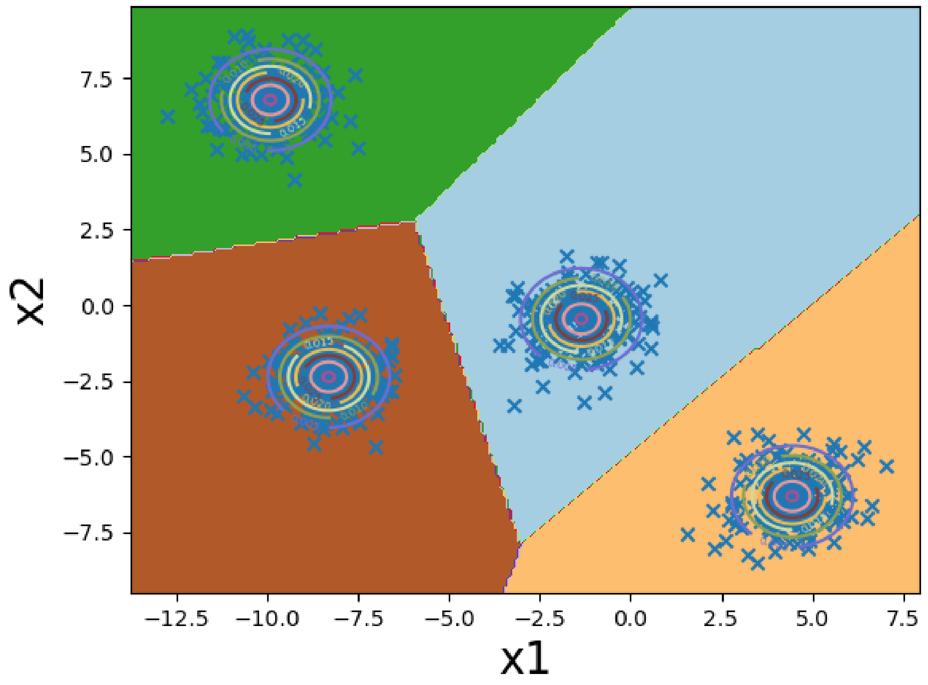
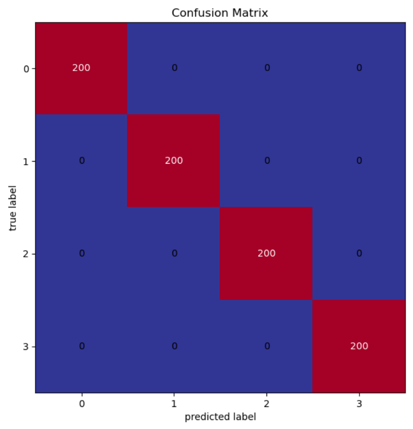


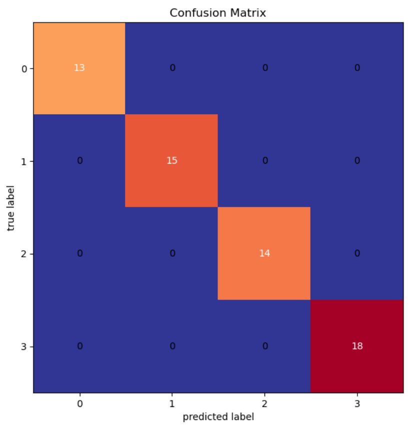
Figure 8: Decision Plot of Naive Bayes Model trained with $C1 = C2 = \sigma^2 I$

I.2.2. Plots for $C1 = C2$



true label \ predicted label	0	1	2	3
0	200	0	0	0
1	0	200	0	0
2	0	0	200	0
3	0	0	0	200

(a) Confusion Matrix for training data



true label \ predicted label	0	1	2	3
0	13	0	0	0
1	0	15	0	0
2	0	0	14	0
3	0	0	0	18

(b) Confusion Matrix for test data

Figure 9: Confusion Matrix for Naive Bayes Model trained with $C1 = C2$

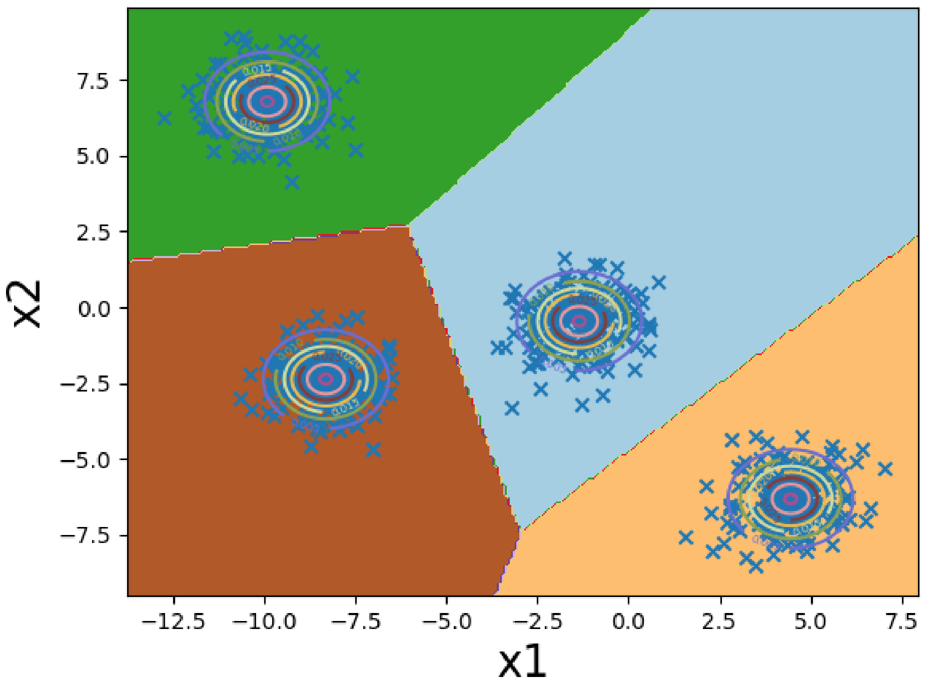
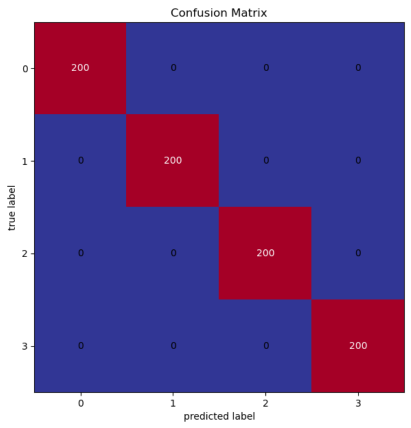
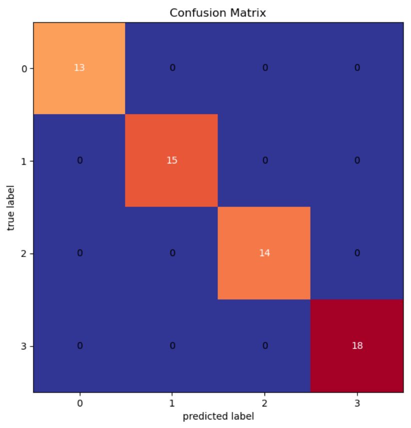


Figure 10: Decision Plot of Naive Bayes Model trained with $C1 = C2$

I.2.3. Plots for $C1 \neq C2$



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 11: Confusion Matrix for Naive Bayes Model trained with $C1 \neq C2$

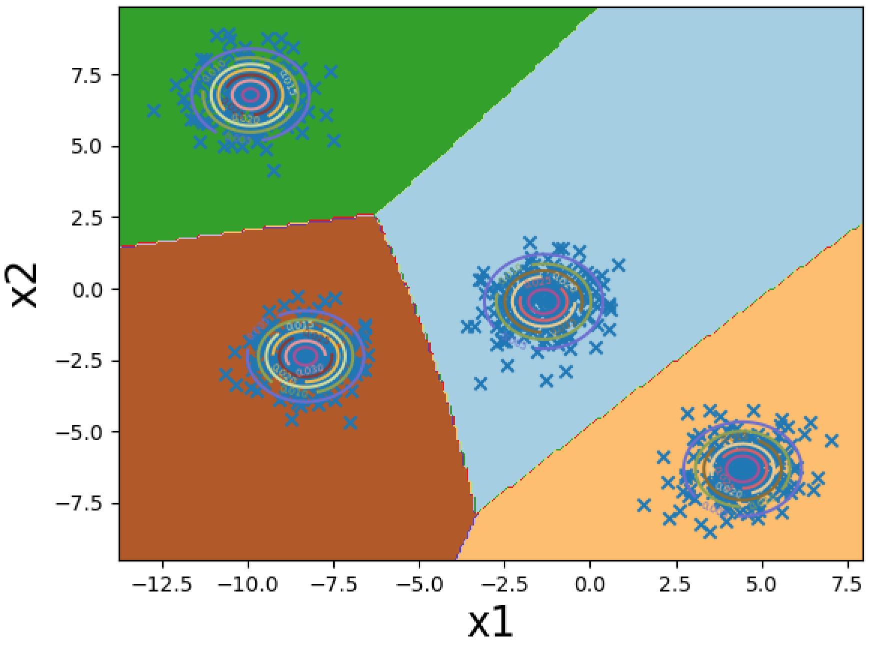


Figure 12: Decision Plot of Naive Bayes Model trained with $C1 \neq C2$

II. Pattern Classification on Non Linearly Separable Data

II.1. K-Nearest Neighbours Classifier

The given synthetic data is classified using KNN Method. The training data is of the shape of a spiral with 3 arcs corresponding to each class. Let us look at some interesting plots obtained for the KNN method for the given dataset.

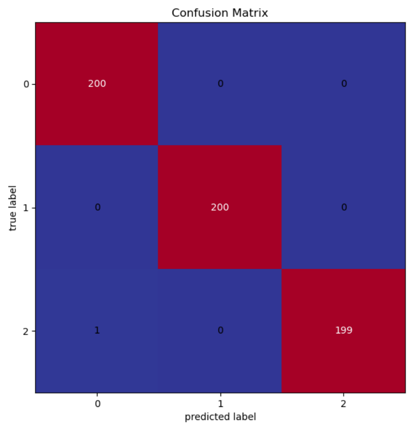
The dip in accuracy of the models in training, validation and test dataset is because of the ambiguity at the end of each arc of the spiral as well as at the centre of it. Otherwise the model seems to classify other data points well and the decision plots give a good representation of them. For *K=7* we get a 100% prediction in both test and validation data. We represent the decision plots for different values of K.

Model	Training Accuracy	Val Accuracy
K=1	99.84%	100%
K=7	99.50%	100%
K=15	99.67%	100%

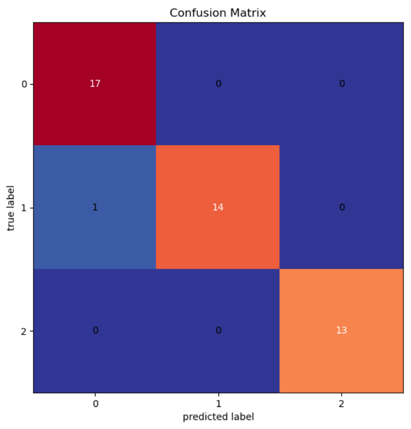
Table 3: Performance of various KNN Models

.

II.1.1. Plots for K=1



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 13: Confusion Matrix for KNN Model trained with K=1

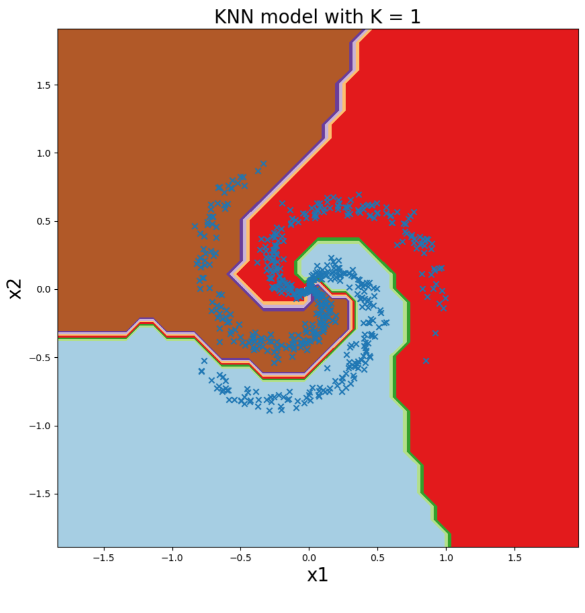
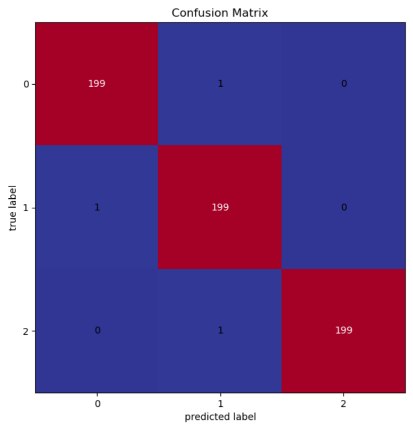
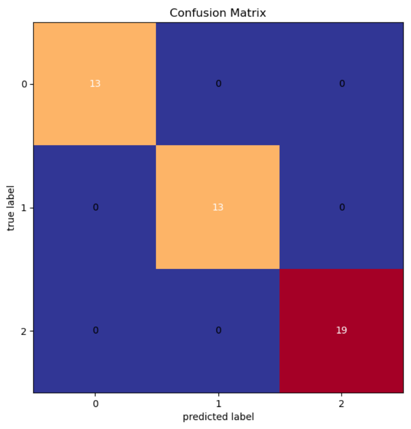


Figure 14: Decision Plot KNN Model trained with K=1

II.1.2. Plots for K=7



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 15: Confusion Matrix for KNN Model trained with K=7

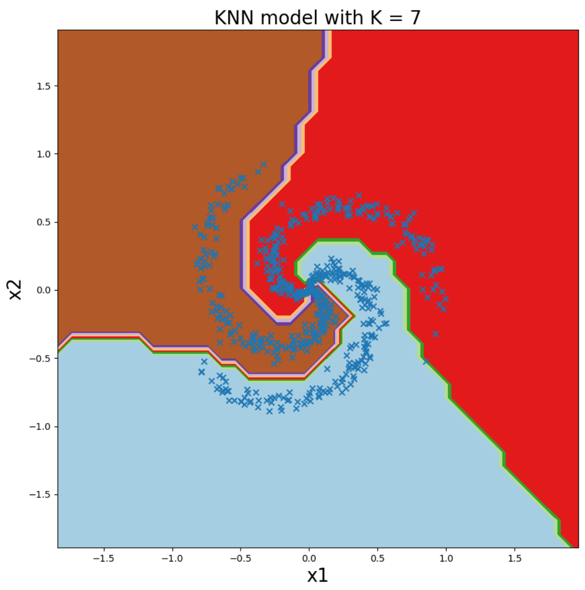
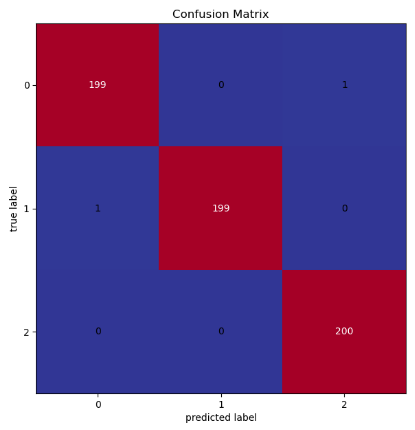
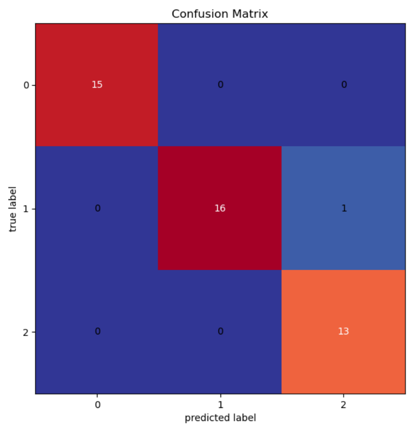


Figure 16: Decision Plot KNN Model trained with K=7

II.1.3. Plots for K=15



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 17: Confusion Matrix for KNN Model trained with K=15

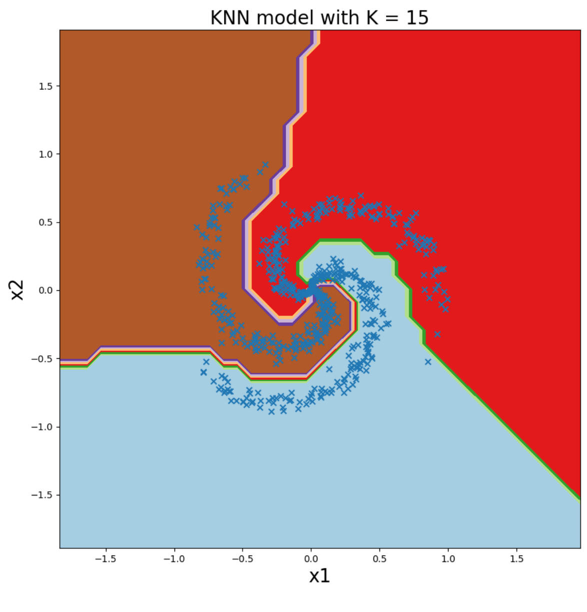


Figure 18: Decision Plot KNN Model trained with K=15

II.2. Bayes Classifier with GMM and diagonal covariance matrix

In this method we represent the density function of a class with weighted normal distributions each trying to capture the non-linearity in the system. Let us look at interesting results for different hyper parameters which in this case is the number of normal functions used.

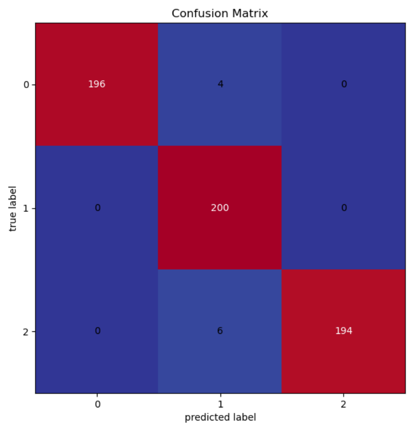
Model	Training Accuracy	Val Accuracy
[1,1,1]	52.17%	60%
[3,3,3]	98.84%	100%
[5,5,5]	98.84%	100%
[7,7,7]	98.84%	100%
[10,10,10]	98.84%	100%
[15,15,15]	99.67%	100%

Table 4: Performance of various Models

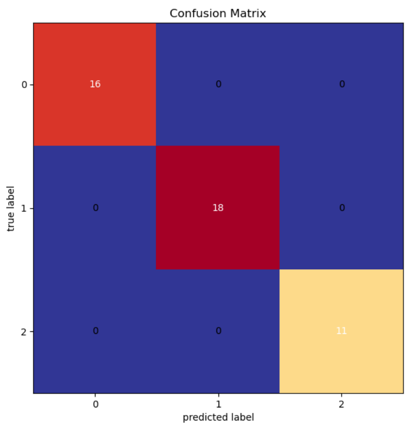
.

In the above table [x,y,z] represents the corresponding number of normal distributions in each class. In our case there are a total of 3 classes. Based on the above table, we can see that the accuracy increases as we increase the number of normal distributions. This means that the non-linearities in each class will now be modelled by more functions and thus the increase in accuracy. Similar to KNN, it has trouble modelling data near the centre of the spiral as well as at the end which accounts for the dip in accuracy in certain cases. We will generalize that our [7,7,7] model to be the good one in this case since it performs slightly better in the val and test set with 100% accuracy in both of them. Other models even though there is a 100% validation accuracy, there was a slight dip in performance in the test set. Let us look at the confusion matrix and decision plot corresponding to that model.

From 20, since we use a diagonal covariance matrix, the level curves of the normal distribution assumed are parallel to the XY axes. The decision region modelled is also completely non linear which shows the power of this particular model as compared to others



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 19: Confusion Matrix for GMM model with diagonal covariance matrix

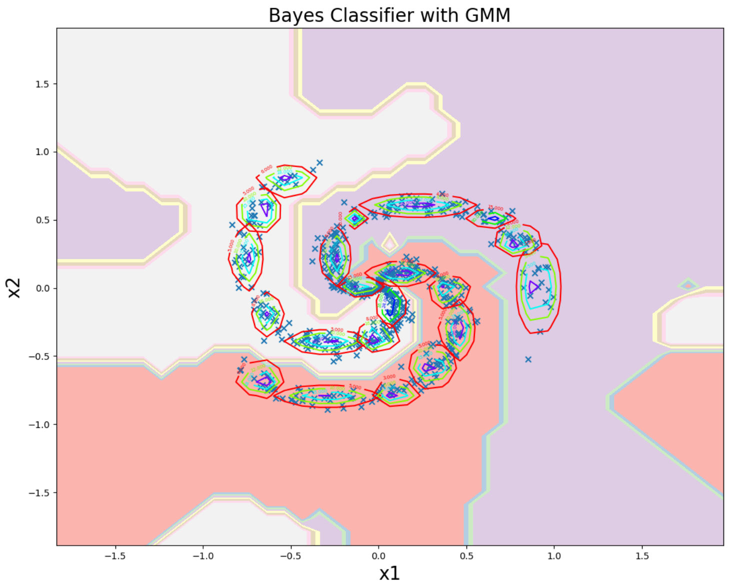


Figure 20: Decision Plot of GMM model with diagonal covariance matrix

II.3. Bayes Classifier with GMM and full covariance matrix

In this method we represent the density function of a class with weighted normal distributions each trying to capture the non-linearity in the system. Let us look at interesting results for different hyper parameters which in this case is the number of normal functions used.

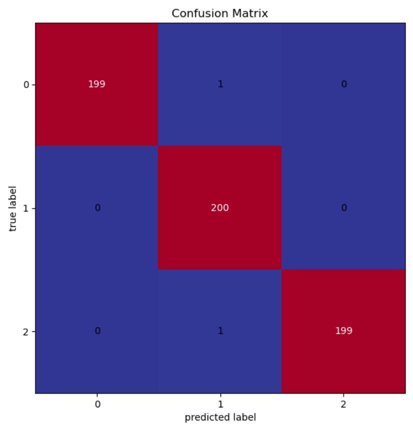
Model	Training Accuracy	Val Accuracy
[1,1,1]	59.17%	66.67%
[3,3,3]	98.84%	100%
[5,5,5]	99.67%	100%
[7,7,7]	99.67%	100%
[10,10,10]	99.67%	100%
[15,15,15]	99.67%	100%

Table 5: Performance of various Models

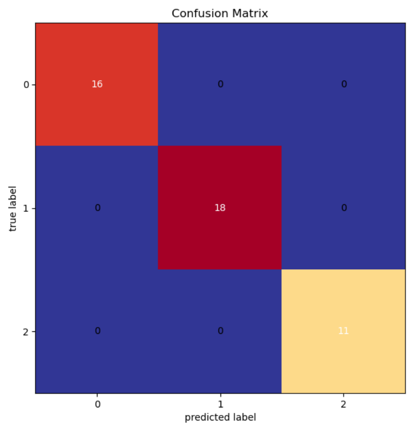
.

In the above table [x,y,z] represents the corresponding number of normal distributions in each class. In our case there are a total of 3 classes. Based on the above table, we can see that the accuracy increases as we increase the number of normal distributions. This means that the non-linearities in each class will now be modelled by more functions and thus the increase in accuracy. Similar to KNN, it has trouble modelling data near the centre of the spiral as well as at the end which accounts for the dip in accuracy in certain cases. We will generalize that our [10,10,10] model to be the good one in this case since it performs slightly better in the val and test set. Let us look at the confusion matrix and decision plot corresponding to that model.

As shown in 22, since we use a full covariance matrix, the level curves of the normal distribution is different than the previous case and now is oriented in a different way. This would mean more number of parameter estimations but better modelling of non linearities. The decision region modelled is also completely non-linear and seems to have a higher non-linearity as compared to the case with diagonal covariance matrix which shows the power of this particular model as compared to others



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 21: Confusion Matrix for GMM model with full covariance matrix

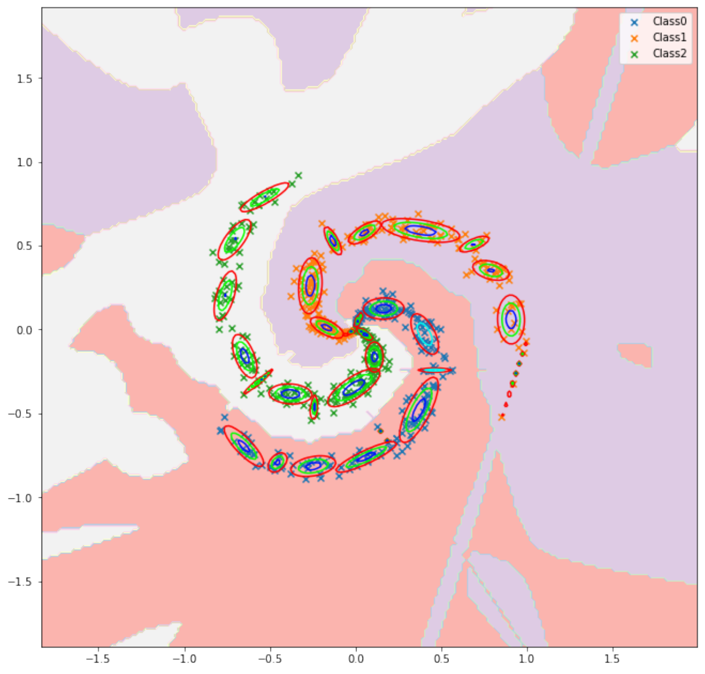


Figure 22: Decision Plot of GMM model with full covariance matrix

II.4. Bayes Classifier with KNN Density Estimation

In this method we represent the density function in a different way as explained in the initial sections. Let us look at some interesting plots.

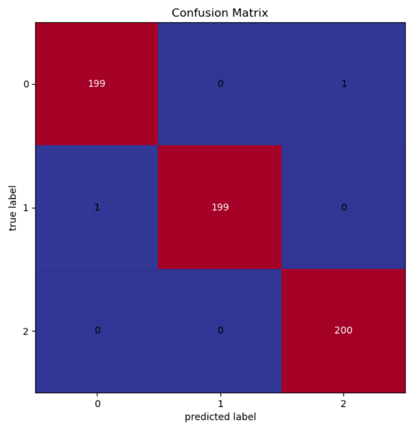
The dip in the accuracy is similar to what was observed in the KNN method. The problem occurs in the end of the arc of the spiral and the centre of it where the method is ambiguous in determining the nearest neighbours in an effective way similar to other models we have seen. The test accuracy was 97.78% in both the cases. The decision surfaces of both the models are plotted. As we increase the value of K we see that the decision boundary fails to cature the end of the spiral of each clas. This may be because, we may now have more data points in the other class which may possibly make the decision boundary move towards them.

Model	Training Accuracy	Val Accuracy
K=10	99.67%	100%
K=20	98.67%	100%

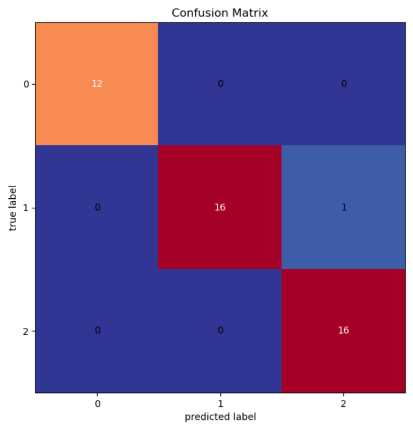
Table 6: Performance of various Models

.

II.4.1. Plots for K=10



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 23: Confusion Matrix with K=10

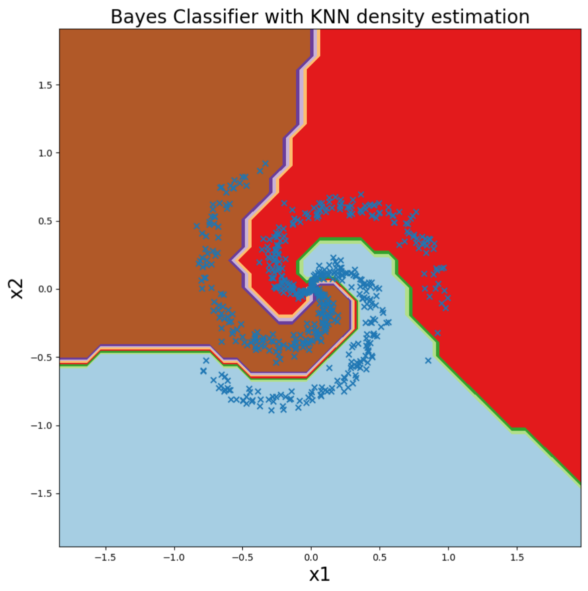
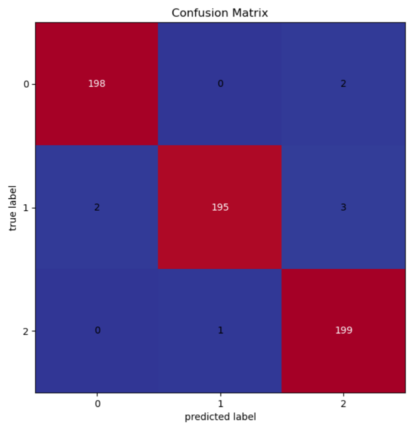
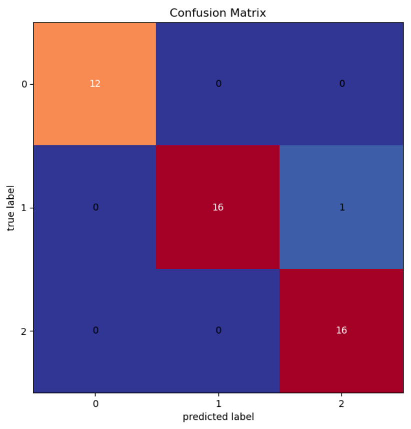


Figure 24: Decision Plot with K=1

II.4.2. Plots for K=20



(a) Confusion Matrix for training data



(b) Confusion Matrix for test data

Figure 25: Confusion Matrix with K=20

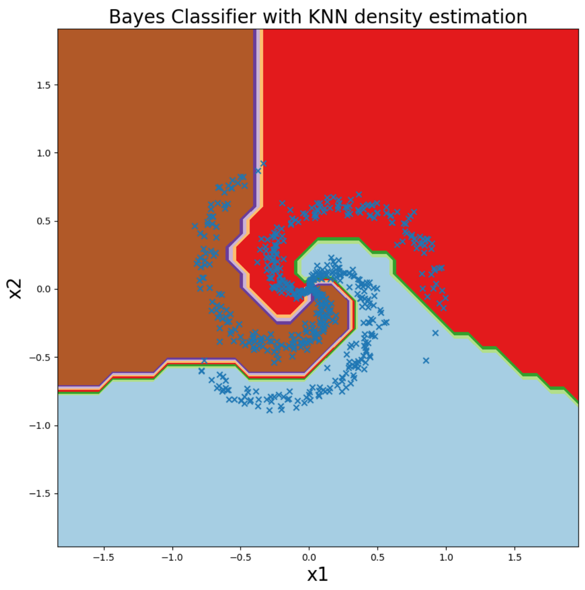


Figure 26: Decision Plot with K=20

III. Static Pattern Classification on Real World Data

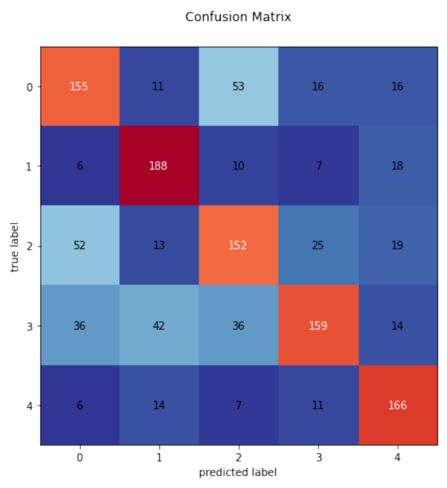
Dataset 2A consists of image histogram data pertaining to five class labels in which each image is represented by a 24 dimensional feature vector. The training accuracies were found to increase with increasing number of clusters in both cases. The best model was found to require much lower cluster numbers of gaussians in the case of full covariance matrix models compared to diagonal covariance matrix models.

The optimal model for diagonal covariance matrix model being [5,5,5,5,5], The test accuracy was 50.87%. The optimal model for full covariance matrix model being [4,4,4,4,4] and the test accuracy was 57.23%.

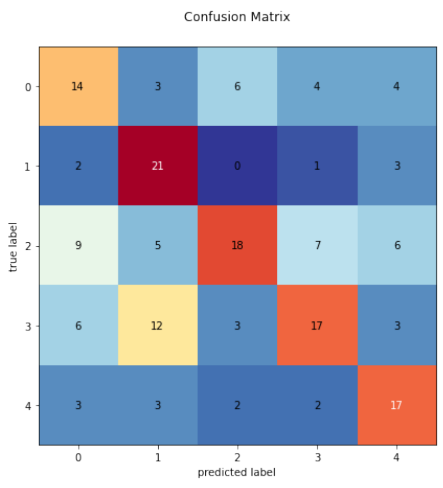
Model	Training Accuracy	Val Accuracy
[1,1,1,1,1]	52.41%	45.90%
[2,2,2,2,2]	56.98%	40.98%
[3,3,3,3,3]	60.30%	46.99%
[4,4,4,4,4]	64.20%	49.18%
[5,5,5,5,5]	66.55%	53.55%
[8,8,8,8,8]	73.78%	51.36%
[11,11,11,11,11]	77.72%	49.18%

Table 7: Performance of models for Diagonal Covariance matrix

.



(a) Train Data



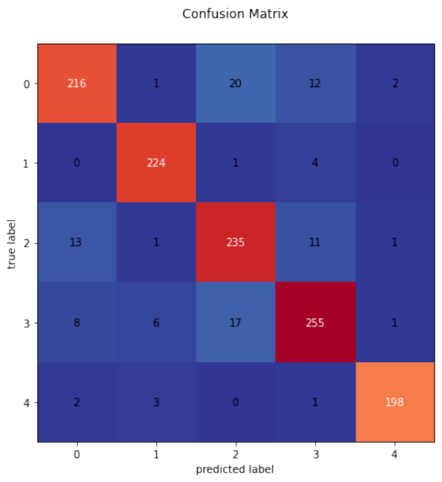
(b) Validation Data

Figure 27: Confusion Matrix for diagonal covariance

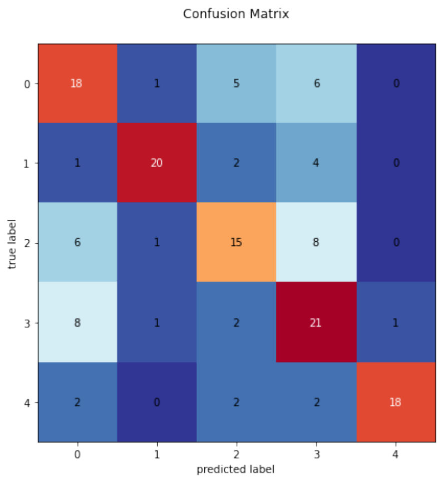
Model	Training Accuracy	Val Accuracy
[1,1,1,1,1]	68.26%	56.28%
[2,2,2,2,2]	76.21%	57.92%
[3,3,3,3,3]	83.44%	58.54%
[4,4,4,4,4]	87.26%	60.0%
[5,5,5,5,5]	91.15%	55.71%
[8,8,8,8,8]	96.27%	51.43%
[10,10,10,10,10]	98.56%	42.86%

Table 8: Performance of models for Full Covariance matrix

.



(a) Train Data



(b) Validation Data

Figure 28: Confusion Matrix for full covariance

IV. Varying Length Pattern Classification for Real World Data

Dataset 2B consists of image histogram data pertaining to five class labels in which each image is represented by a set of 36, 23 dimensional vectors, i.e each image is represented by $36x23$ matrix. In the training phase each of row from the image matrices are treated as separate examples with the same label as the parent image. In the classification phase, the classification of each image involves the calculating of the conditional probability that the image belongs to the class i as the product of the individual conditional probabilities of its constituent rows, i.e for an image matrix X and class label i.

$$\begin{aligned} p(y = y_i) &= \prod_{n=1}^{n} p(y = y_i) \\ &= \prod_{j=1}^{N} \sum_{k=1}^{K} w_{ik}.\mathcal{N}(x_n|\mu_{ik}, c_{ik}) \end{aligned}$$

In both cases, the optimal model was found to be the cluster combination of [8,8,8,8,8] and a thorough search through the combination parameters were not feasible as the training of the model is computationally expensive across 150,000 data points

Model	Training Accuracy	Val Accuracy
[1,1,1,1,1]	23.13%	22.88%
[2,2,2,2,2]	45.77%	47.17%
[3,3,3,3,3]	57.62%	47.45%
[4,4,4,4,4]	61.85%	51.97%
[5,5,5,5,5]	67.37%	61.58%
[6,6,6,6,6]	69.64%	62.42%
[7,7,7,7,7]	70.21%	66.10%
[8,8,8,8,8]	73.62 %	66.66 %

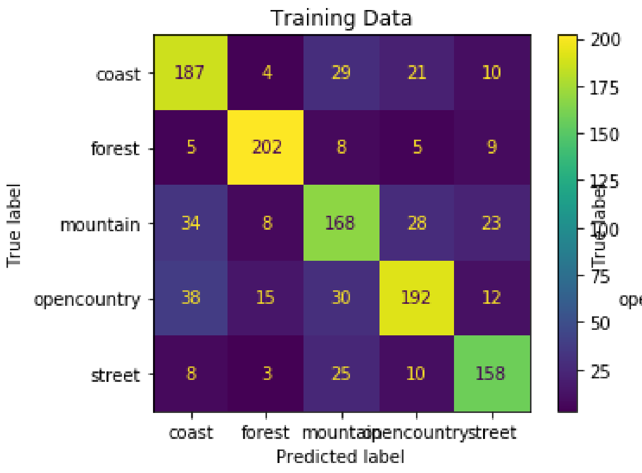
Table 9: Performance of models for Diagonal Covariance matrix

.

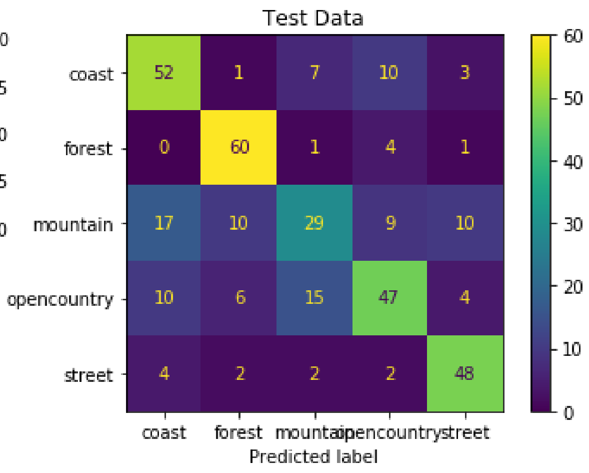
Model	Training Accuracy	Val Accuracy
[1,1,1,1,1]	88.39%	83.61%
[2,2,2,2,2]	94.80%	80.50%
[3,3,3,3,3]	95.86%	85.31%
[4,4,4,4,4]	97.40%	90.67 %
[5,5,5,5,5]	98.86%	94.63%
[6,6,6,6,6]	99.02%	94.63%
[7,7,7,7,7]	99.26%	94.35%
[8,8,8,8,8](optimal)	99.67 %	95.19 %

Table 10: Performance of models for Full Covariance matrix

.

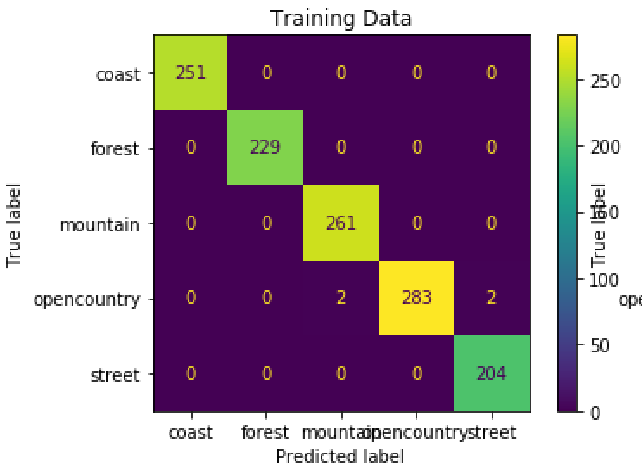


(a) Train Data

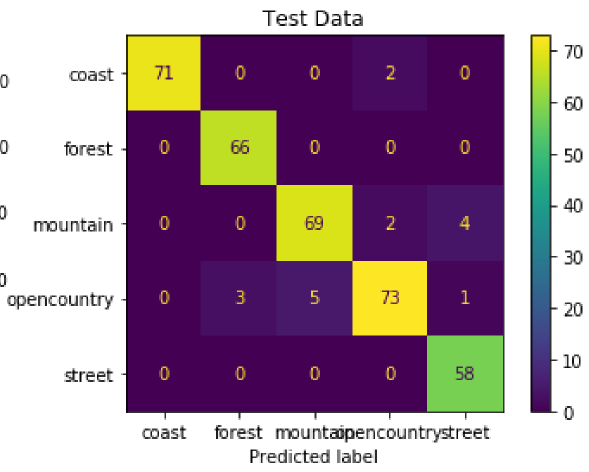


(b) Validation Data

Figure 29: Confusion Matrix for diagonal covariance



(a) Train Data



(b) Validation Data

Figure 30: Confusion Matrix for full covariance