Pattern Recognition and Machine Learning

Lab Assignment 3

Name-Akriti Gupta
Roll Number-B21AI005

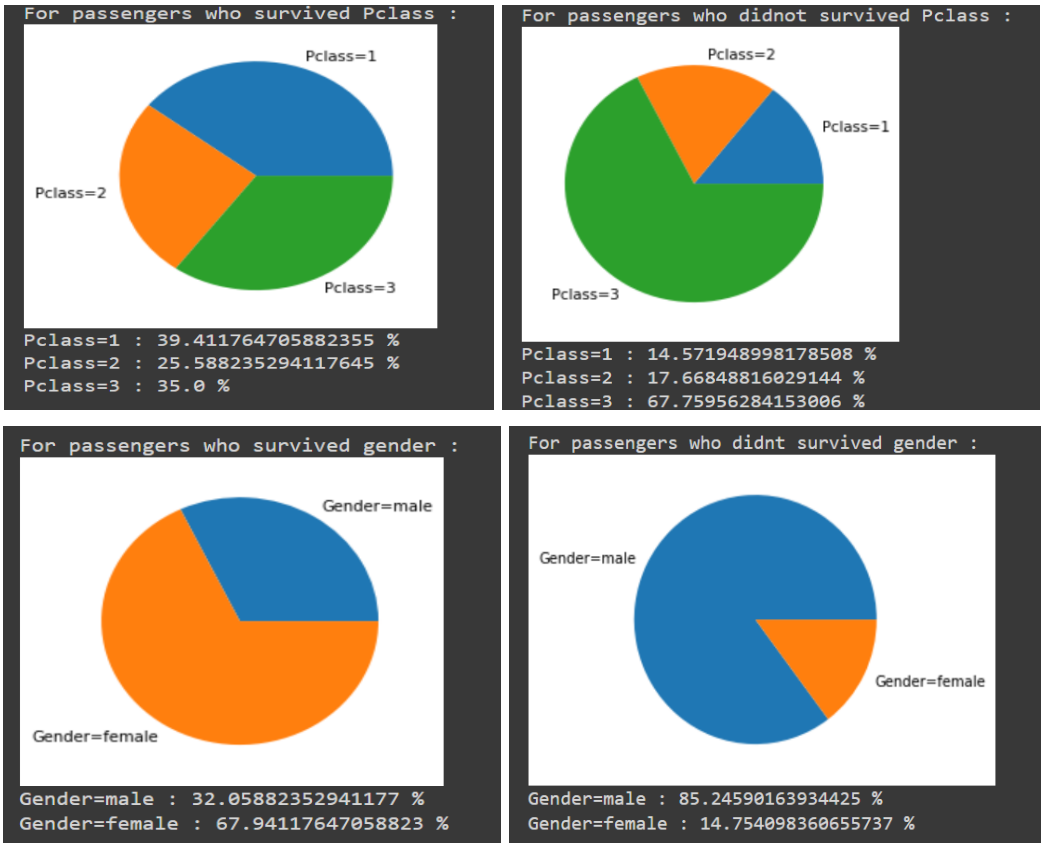
Question 1

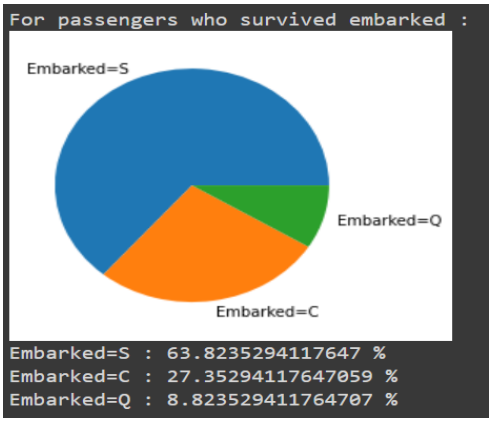
1. Perform pre-processing and visualization of the dataset. Split the data into train and test sets. Also identify the useful columns and drop the unnecessary ones.

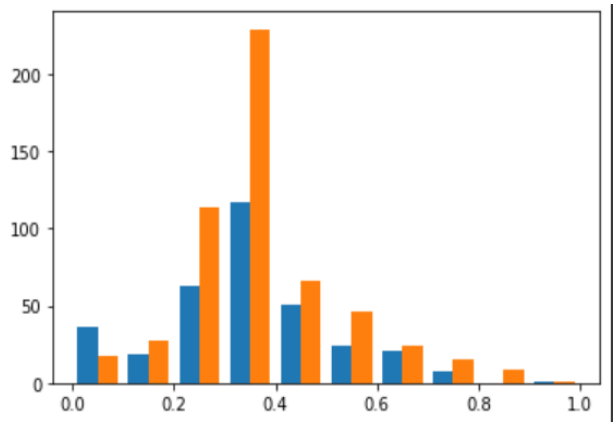
For pre-processing we first drop the clearly visible irrelevant columns ie. which are not related to whether the passenger survives or not. So, we drop the Name, PassengerId and Ticket column.
Now, we analyze for NaN values in our dataframe and get the following:

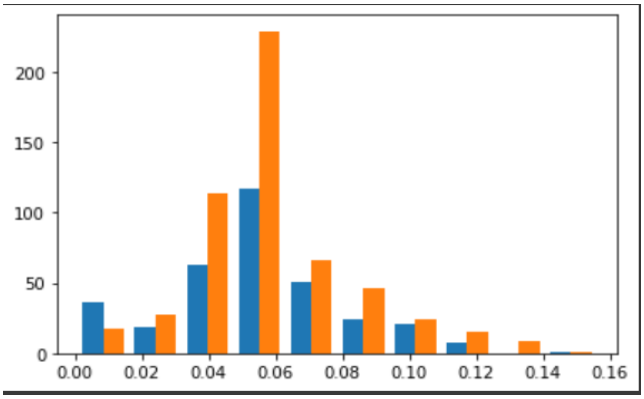
```
Pclass        0
Sex           0
Age         177
Fare          0
Cabin       687
Embarked      2
Survived      0
dtype: int64
```

So, we observe that there are more NaN values in the Cabin column than there are in non NaN values in it so, considering the fact of shortage of appropriate amount of data in the Cabin column, we drop it.
For the Age column we fill the NaN values with the mean of the non NaN values of Age column. Now, we have only 2 NaN values which is very less compared to the size of data, so without fear of loss of data we drop the NaN values.
Now for the categorical features(Sex and Embarked) as they are nominal features so we encode them to integral values according to what their value is.
Now, we scale the higher ranged features(Age and Fare) to range of 0 to 1 ie. apply MinMaxScaler so that they are in the same scale as other features.
For visualizing the data we plot pie graphs for the categorical features(% of survived for each feature labeled value) and for continuous features we plot histogram for the same. The plots are:









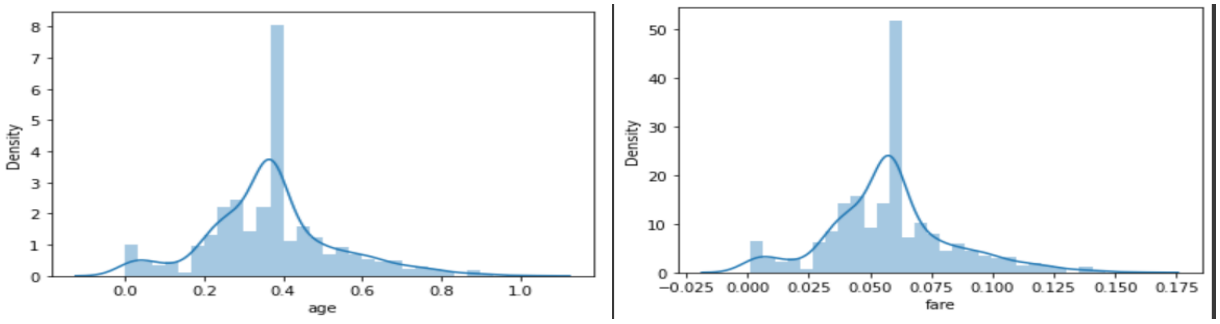
Now we split our data into training and testing with training size as 70%.

2)Identify the best possible variant of naive bayes classifier for the given dataset. Justify your reason for the same.

We know that gaussian naive bayes work good when there are continuous features in our dataset and if the continuous features follow a nearly normal curve then gaussian naive bayes would definitely be the best model to implement.
Here we have 3 possible variants of Naive Bayes:
i)Gaussian Naive Bayes
ii)Multinomial Naive Bayes
iii)Bernoulli Naive Bayes
We fit an object of each class and look at the score the models give on testing data,

```
Training Score for gaussian model : 92.26190476190477%
Testing Score for gaussian model : 92.85714285714286%
Training Score for multinomial model : 85.71428571428571%
Testing Score for multinomial model : 88.09523809523809%
Training Score for bernoulli model : 69.77491961414791%
Testing Score for bernoulli model : 71.16104868913857%
```

So, from the score it is clear that gaussian model has the best score, we also check if the continuous features follow normal distribution or not



So, we also observe that the continuous features follow normal density distribution.
Finally calculating the f1_score confirms us that the Gaussian variant of Naive Bayes will be the best model for our purpose.

3)Implement the identified variant of Naive Bayes Classifier using scikit learn,report its performance based on appropriate metrics.(ROC AUC etc)

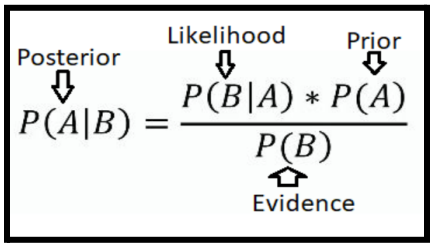
For implementing the identified Gaussian Naive Bayes model, we make a class GaussianNaiveBayes which we implement from scratch
The class GaussianNaiveBayes has the following methods:
i)get_prior_probability( ):The function computes and returns prior probability of a class(given as parameter to function)

ii)categorical_likelihood( ):The function calculates likelihood/class conditional probability of a point wrt a certain class_label passed as argument. We use this to compute class conditional probabilities for categorical features.

iii)gaussian_likelihood( ):The function also calculates likelihood/class conditional probability of a point wrt a certain class_label passed as argument. We use this to compute class conditional probabilities for continuous features by assuming gaussian/normal distribution. Assuming the independence assumption

of naive bayes we compute the probability as the product of individual probabilities.

iv)get_posterior_probability_point( ):The function returns posterior probabilities wrt all the class labels for a datapoint.
We use the following formula to compute the posterior probability:



v)predict_point( ):The function obtains the posterior probability of both classes from the get_posterior_probability_point( ) function and returns the class for which we have greater value of posterior probability.

vi)predict( ):The function takes the x_test array as input and returns the predicted class as calculated following the theory of gaussian naive bayes.

vii)predict_proba( ):The function returns array where each element represents the posterior probability of the respective class for that point.

viii)plot_roc( ):The function plots the ROC curve wrt the model trained.

ix)get_auc_score( ):The function returns the area under the curve for the ROC curve of the model.

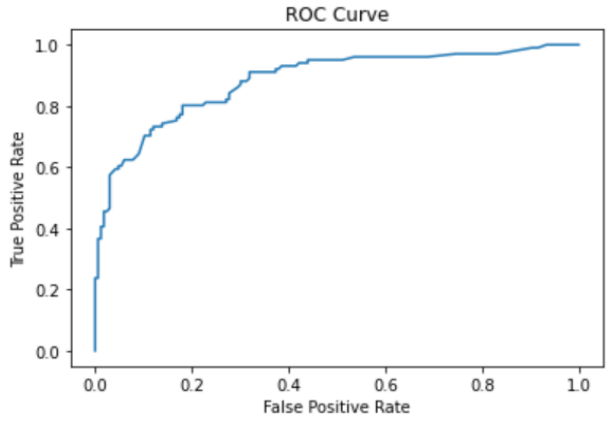
x)get_metrics( ):The function returns a tuple of (covariance matrix, precision, recall, f1 score)

xi)score( ):The function returns the score of our model trained.

Thus using the above functions we obtain the y_predicted(of testing data) as:

```
y predicted array :
[1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0
 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0
 1 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0
 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 1 1
 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1
 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 0 0 0 1
 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1
 0 1 0 0 0 1 0 0]
```

Now, we plot the ROC curve using plot_roc( ) method and get the plot:



And also obtain the area under the AUC curve as : `0.8833949660026243`
We, obtain the other performance metrics(like precision, confusion matrix,f1 score) from the get_metrics( ) method

Confusion Matrix :
[[143 23]
 [26 75]]

Precision :
76.53061224489795

Recall :
74.25742574257426

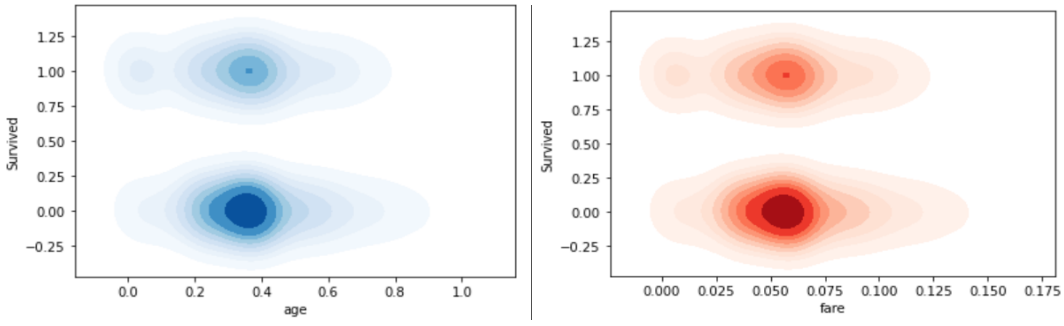
F1 Score :
75.37688442211056

4)Perform 5 fold cross validation and summarize the results across the cross-validation sets. Compute the probability of the top class for each row in the testing dataset.

For 5 Fold cross validation we implement it from scratch by splitting the data into 5 parts and each time 4 used as training to train the model and one part to test our results upon and obtain the cross validation score as: `78.87096774193549%`
For computing probability of top class we find the class whose posterior probability is maximum amongst all the data points and then that is only the probability of top class.

5) Make contour plots with the data points to visualize the class-conditional densities. What can you say about the assumption Naive Bayes model is based on from these plots? Explain in your report

For Contour plots we use the function kdeplot( ) from seaborn and get the following plots wrt our 2 continuous features



We, can clearly see that the plots comes out to be nicely separated and hence we can say that assumptions Naive Bayes is based upon is that the features are independent of each other.

6)Compare your model with the Decision Tree classifier on the same dataset by performing 5-fold cross-validation and summarizing the results. Justify why one of them works better on this numeric dataset.

Firstly, we train our DecisionTreeClassifier model and fit the training data into it and then obtain the cross validation score from the scratch function we implemented in part 4 of this question and get the cross validation score to be `77.9032258064516%` which is lesser than that obtained from our gaussian naive bayes classifier, `78.87096774193549%`

The reason for Gaussian Naive Bayes classifier working better in this case are:

i)Our continuous features were nearly following the gaussian distribution and thus approximating it as gaussian density distribution and applying gaussian naive bayes gave us better results.

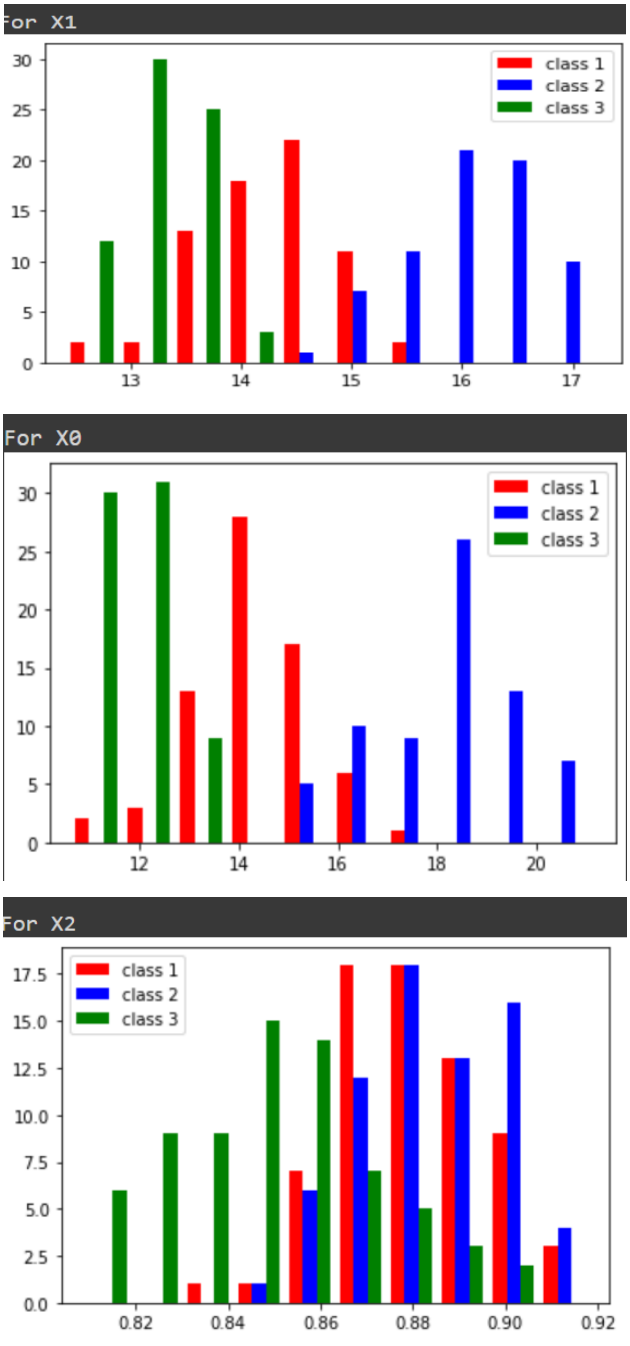
ii)Decision Trees work well for large datasets as it needs more training examples to get better at predicting whereas naive bayes can perform well even on smaller sized samples.

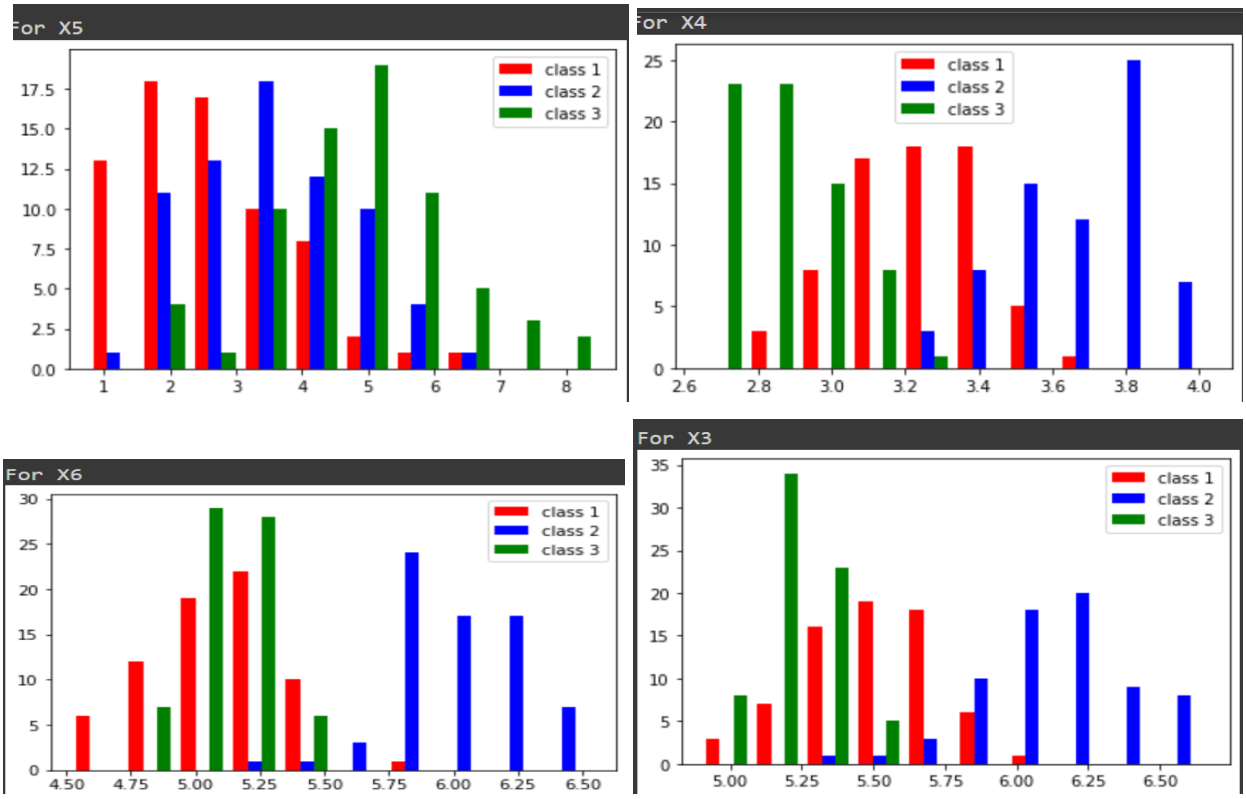
iii)Naive Bayes is a generative model so it works better than DecisionTree and it also works better on categorical data.

Question 2

1)Use histogram to plot the distribution of samples.

The following are the histograms we plot:





2)Determine the prior probability for all the classes.

We obtain prior probabilities as :

Class 1:0.33
Class 2:0.33
Class 3:0.33

3)Discretize the features into bins from scratch. Use of pandas, scikit learn and scipy is not allowed for this subpart.

We make 5 bins with equal intervals and for each of the features encode the feature value according to the bin index and thus get a discretized features.
The new data frame appears as follows:

New Discretized dataframe :

	x0	x1	x2	x3	x4	x5	x6	y
0	2	2	2	2	2	0	1	1
1	2	2	3	1	2	0	1	1
2	1	1	4	1	2	1	0	1
3	1	1	3	1	2	0	0	1
4	2	2	4	2	3	0	1	1
..	..	..	..	..	..	..	..	..
205	0	0	3	0	1	1	0	3
206	0	0	1	0	0	2	1	3
207	1	1	3	0	2	4	1	3
208	0	0	1	0	0	1	1	3
209	0	0	2	0	1	3	1	3

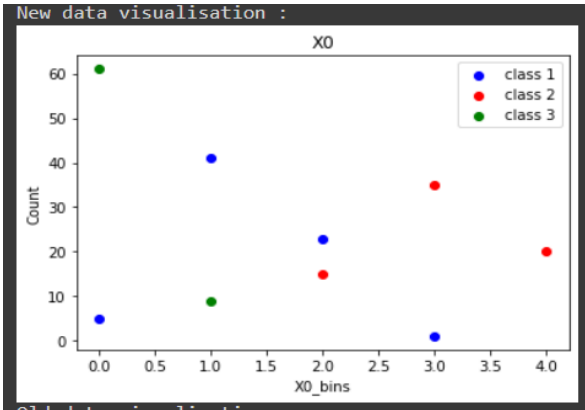
4)Determine the likelihood/class conditional probabilities for all the classes.

For this question we make a similar class as we made in Question 1 subtask 3 With the only difference that this time all the features are made categorical via binning and thus we only use categorical_likelihood of that class and this new class is named NaiveBayes. So using the method get_likelihood( ) we obtain the likelihood for each data point wrt class labels.

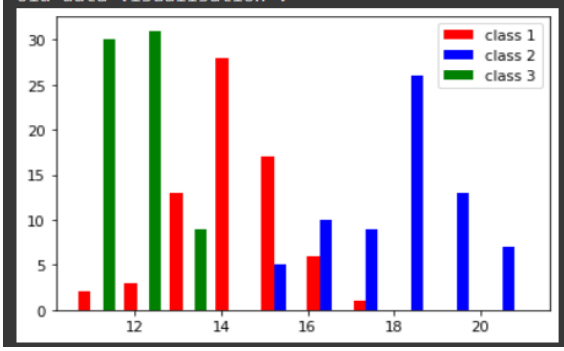
5) Plot the count of each unique element for each class. Compare the plot with the plot of distribution.

Upon plotting the count of each unique feature labels wrt class along with the distribution plots we get the following resulting plots:

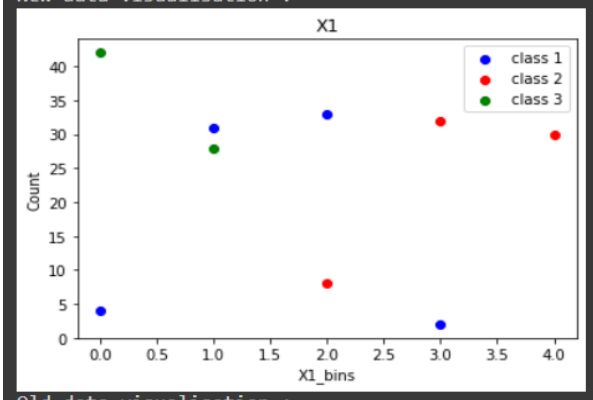
New data visualisation :



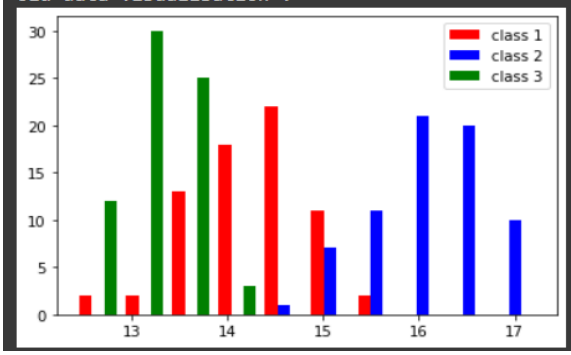
Old data visualisation :



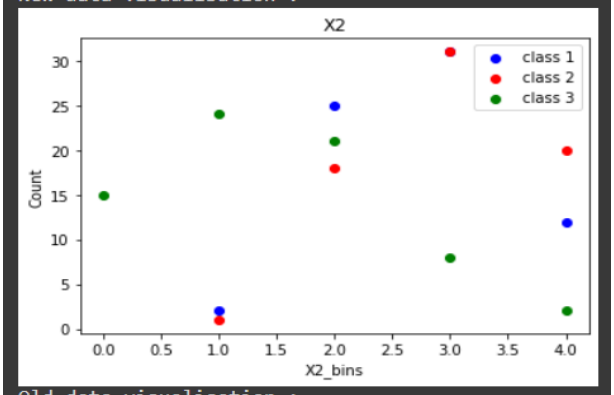
New data visualisation :



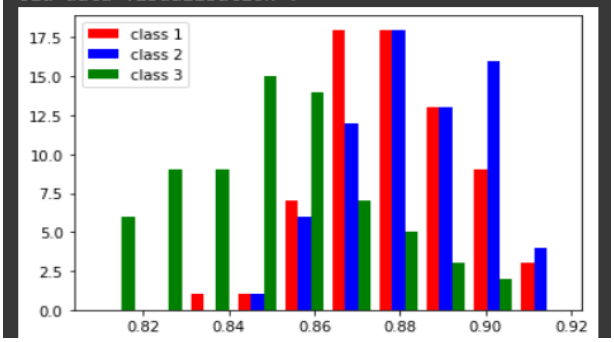
Old data visualisation :



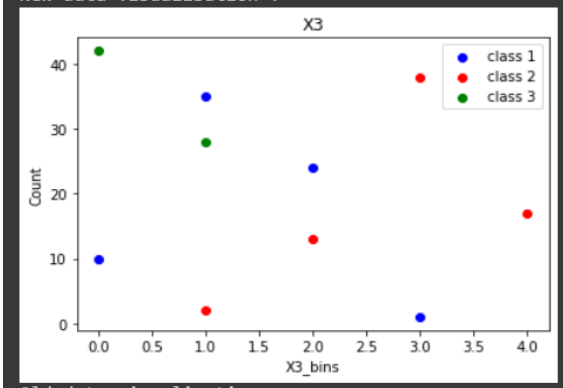
New data visualisation :



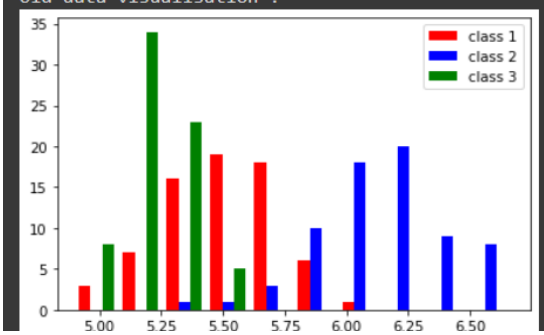
Old data visualisation :

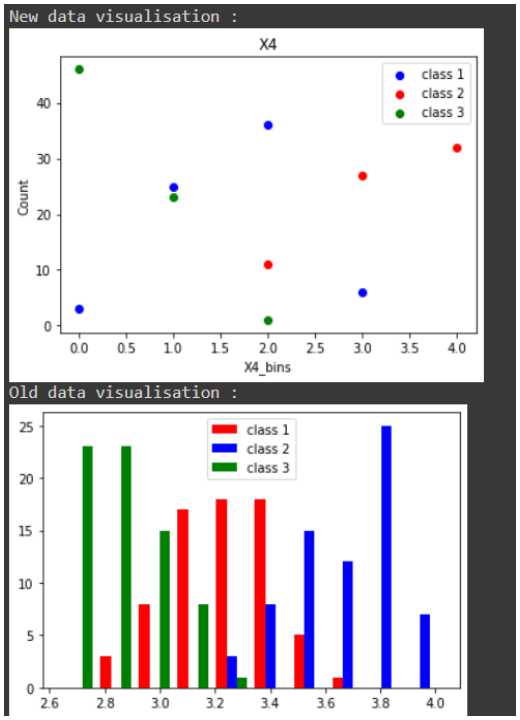


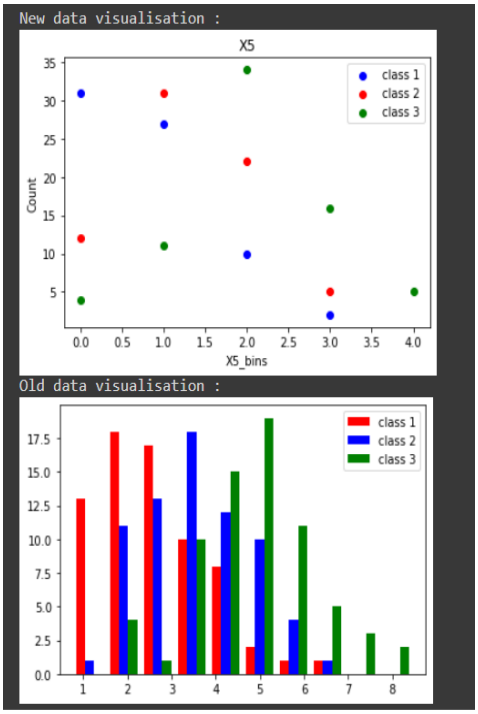
New data visualisation :

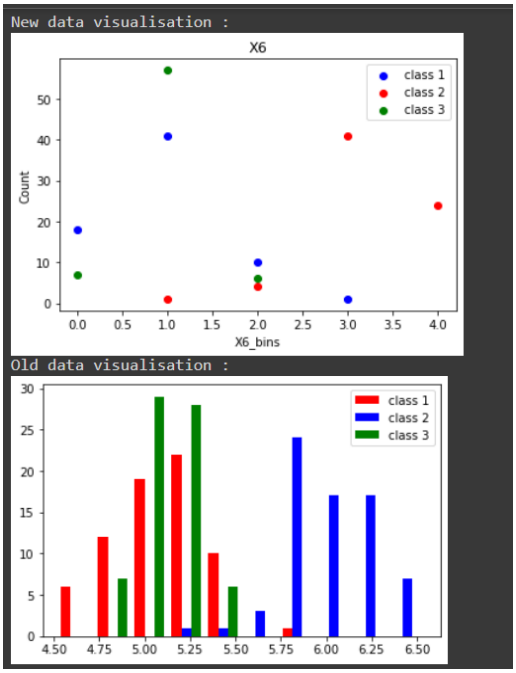


Old data visualisation :





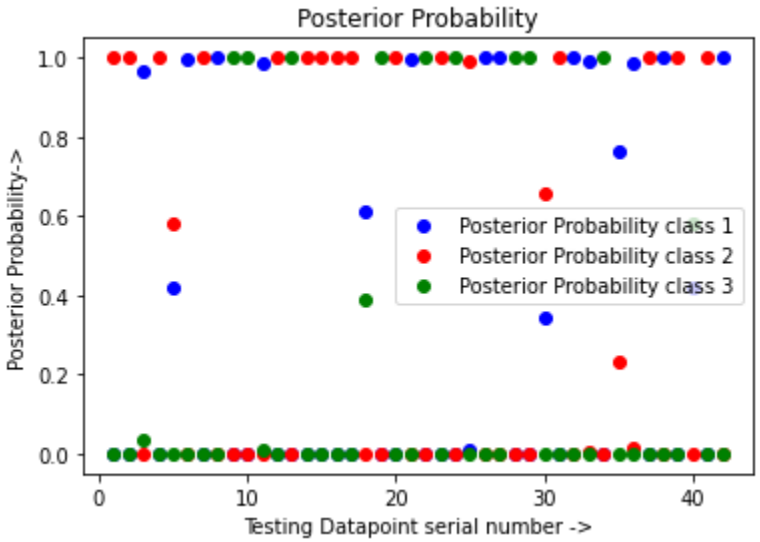




Simultaneously visualizing the plots together, we can clearly see the frequency/count distribution class wise over the bins is nearly the same ie. initially if there were many points of class 1 for lower values of features then finally also we have more count of class 1 in the respective bins.

6)Calculate the posterior probabilities and plot them in a single graph. Analyze the plot.

The posterior probability we obtain from the method predict_proba( ) of our NaiveBayes class and then we plot the posterior probabilities class wise
We get the following plot:



From the posterior probability plot we can say our model predicts the class of a datapoint with very high confidence as the probabilities of top class are mostly 0.999 types and also we obtain the score as `95.23809523809523%`
So, we can say that our model is a very good model with high accuracy/score and with high confidence in predicting the correct class.