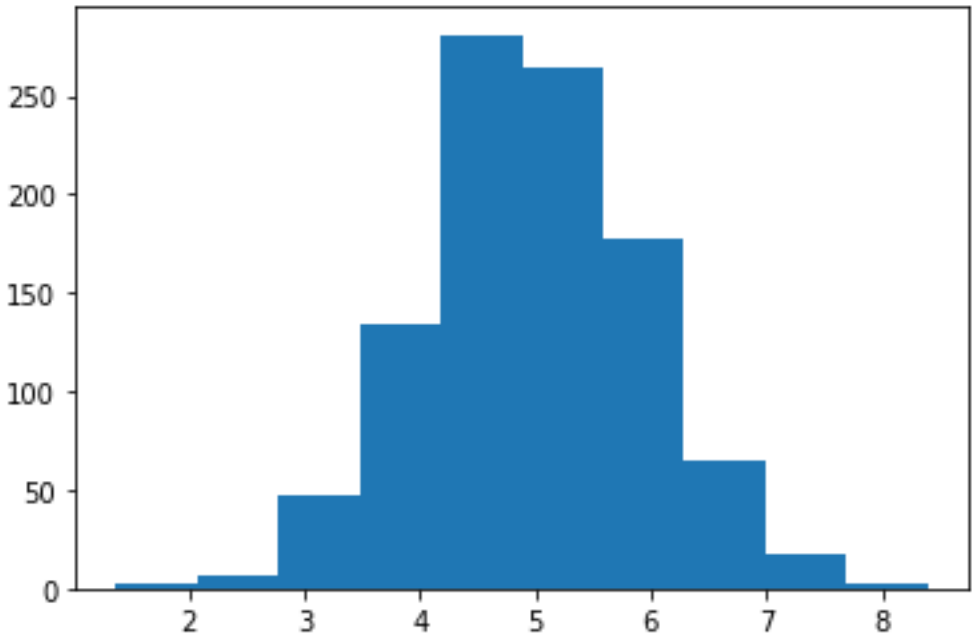
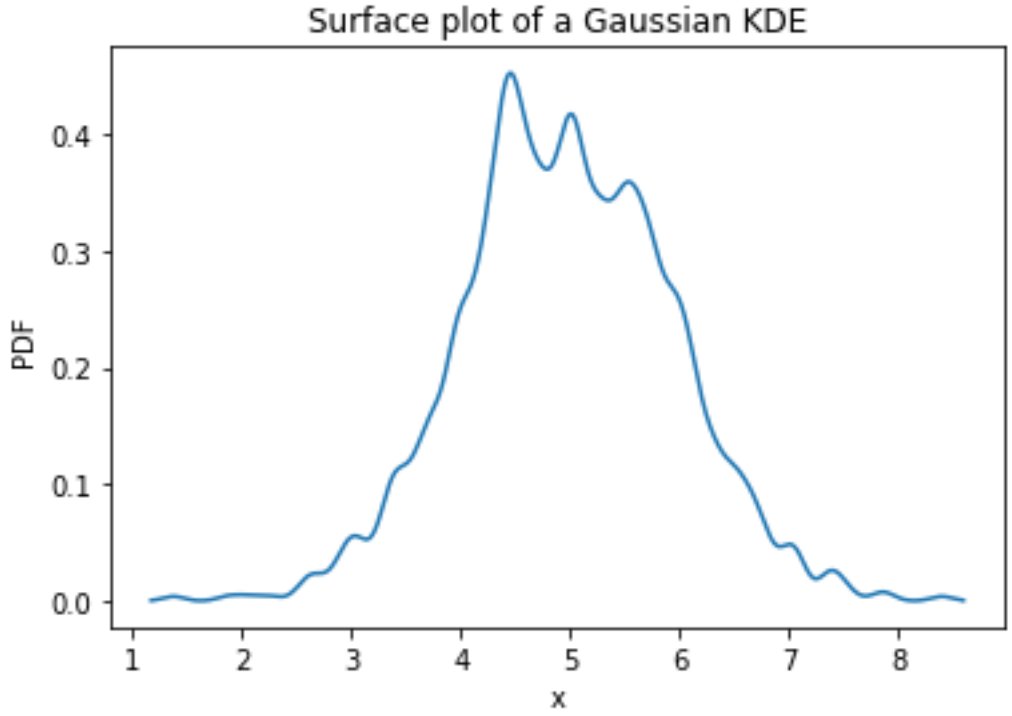
In the first problem we have computed the KDE (Kernel Density Estimation) using Python. It is a non-parametric way to estimate the probability density function of a variable. Then we have proceeded with the question and listed them in this report according to the instructions given in the problem.
In the second problem, we have implemented naïve bayes classifier using Python . It is a classification technique based on Bayes theorem with an assumption of independence predictors. We have proceeded with the question and listed the results as per the instructions given in the problem.

Problem 1.1

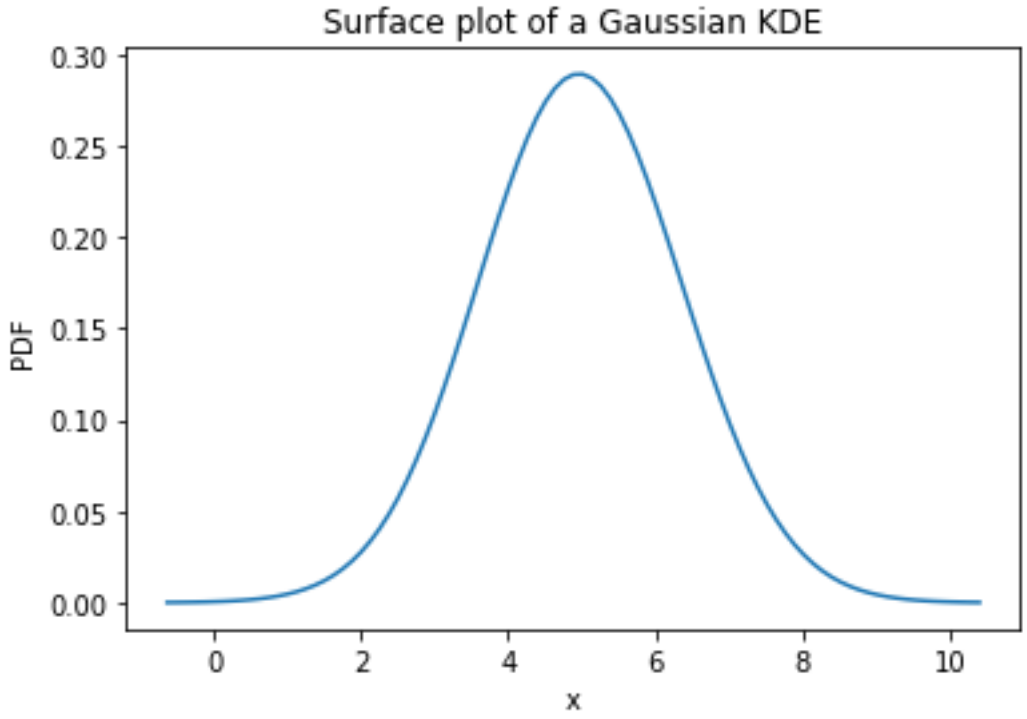
a) Dataset with N = 1000, mean = 5 and std = 1



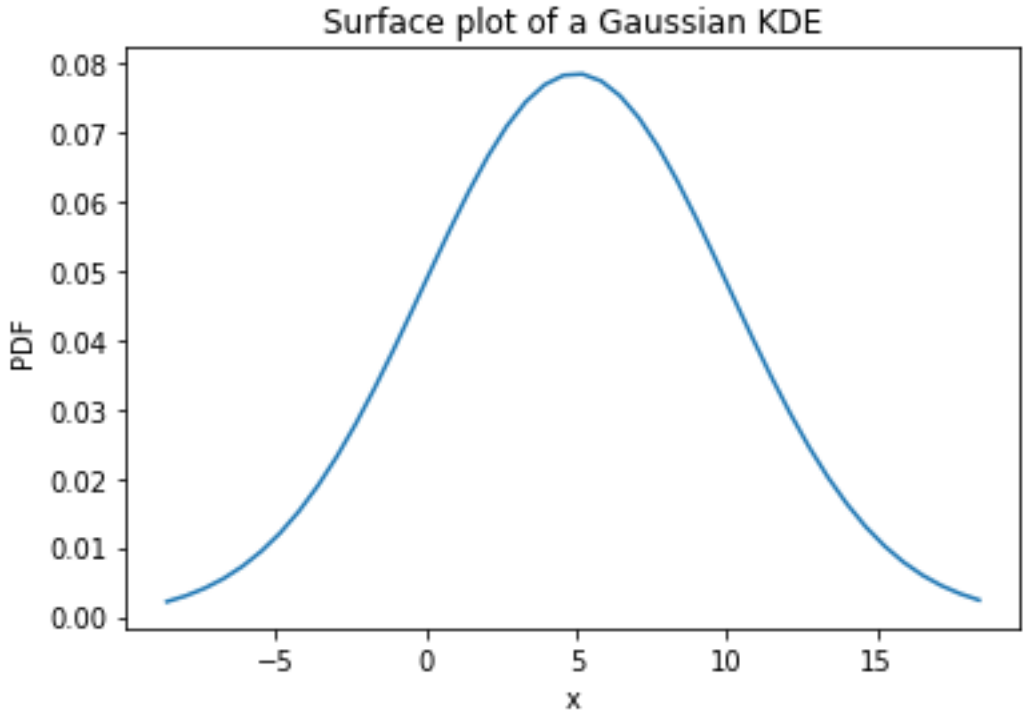
The domain of x-axis = [1.1720208128777057, 8.59691613820467]
The bandwidth = 0.1



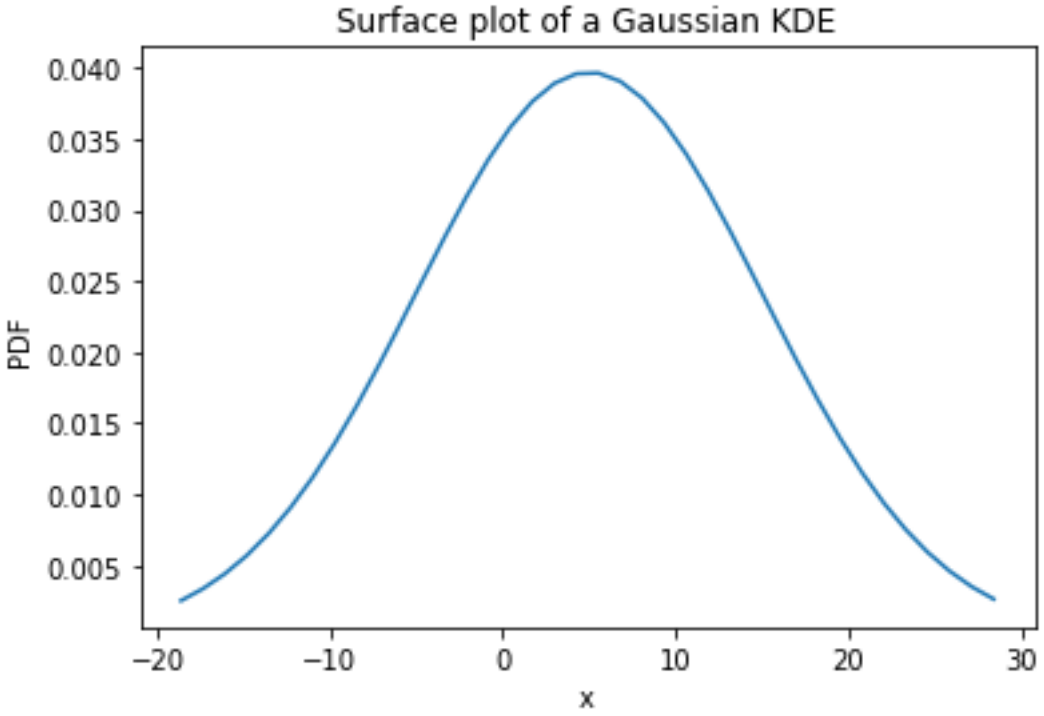
The domain of x-axis = [-0.6279791871222944, 10.39691613820467]
The bandwidth = 1



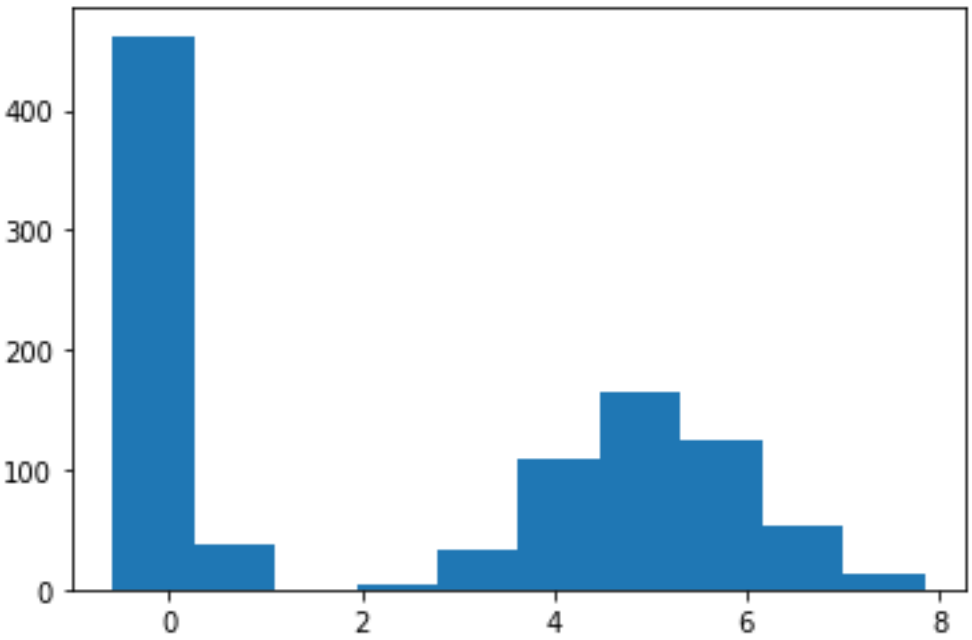
The domain of x-axis = [-8.627979187122294, 18.39691613820467]
The bandwidth = 5



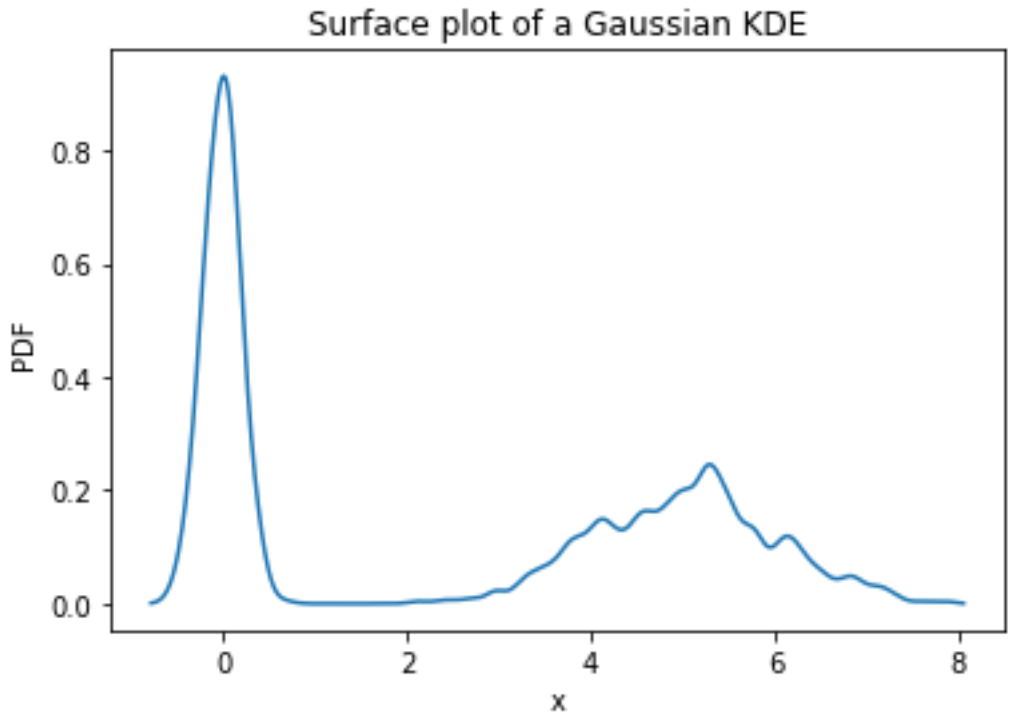
The domain of x-axis = [-18.627979187122293, 28.39691613820467]
The bandwidth = 10



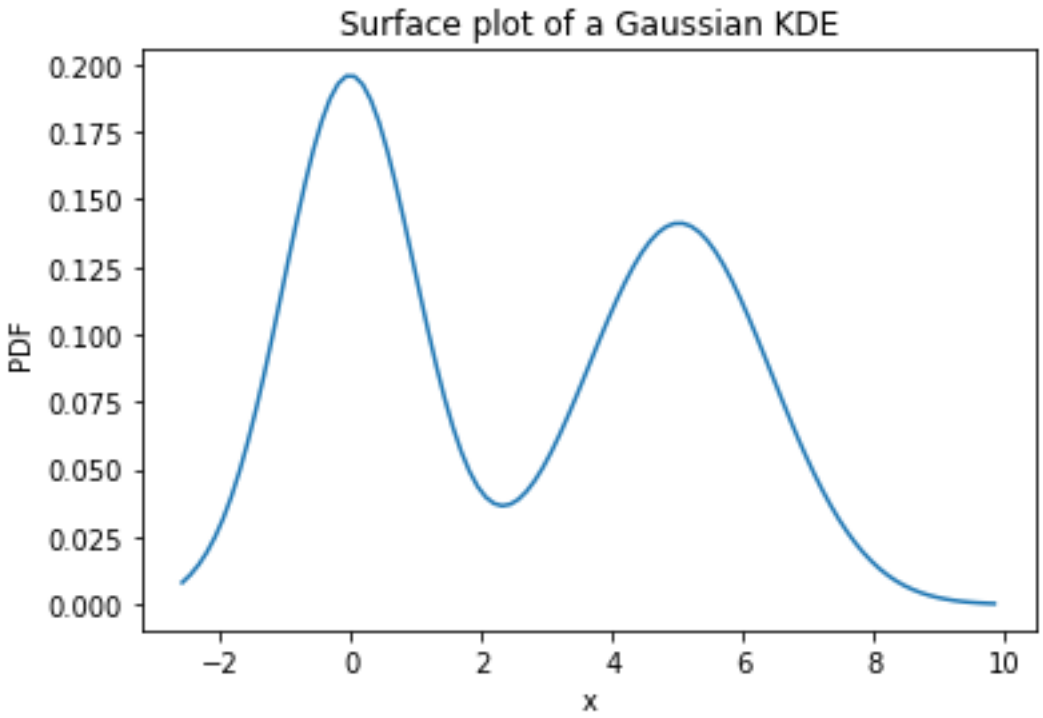
b) Dataset with N = 1000, mean_1 = 5 and std_1 = 1 & mean_2 = 0 and std_2 = 0.2



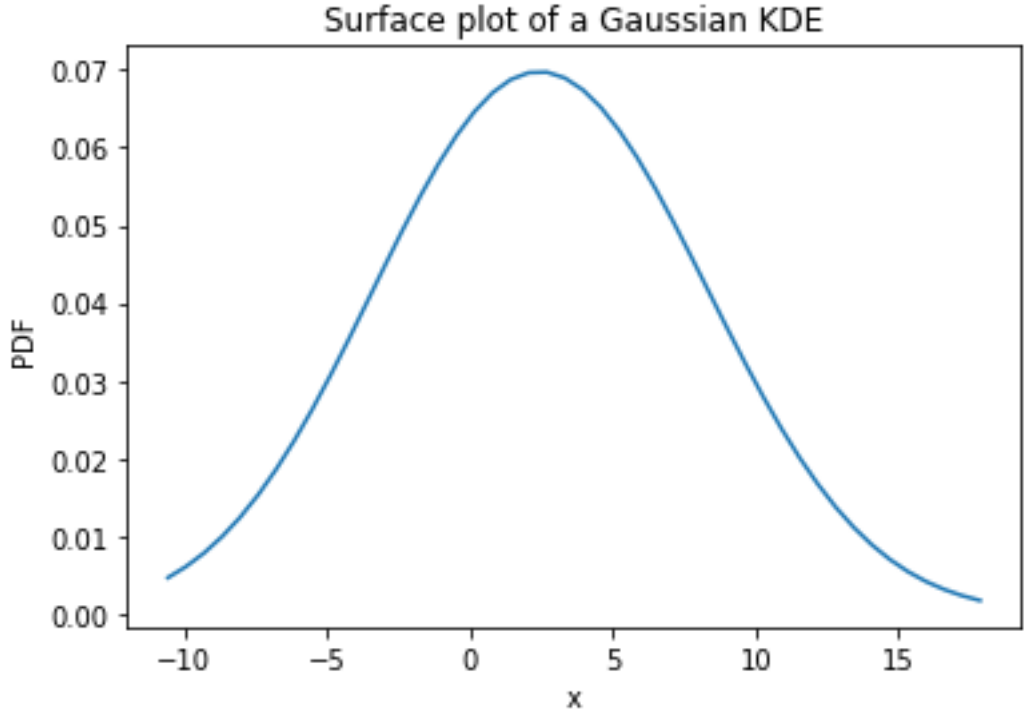
The domain of x-axis = [-0.7851858019277349, 8.050344644359088]
The bandwidth = 0.1



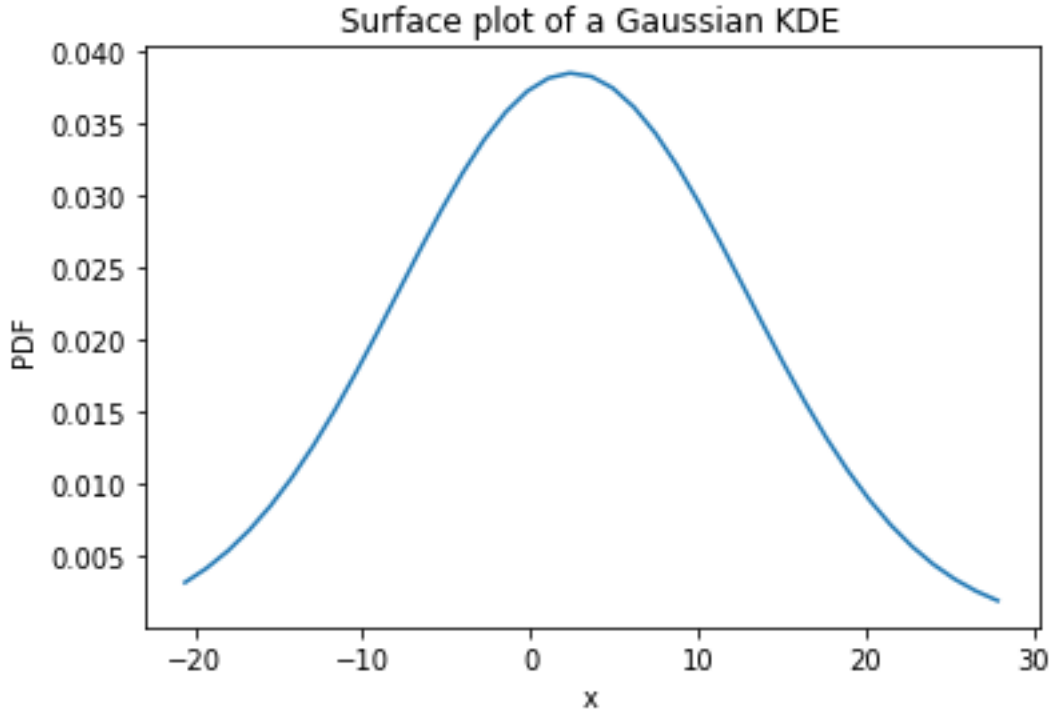
The domain of x-axis = [-2.585185801927735, 9.850344644359089]
The bandwidth = 1



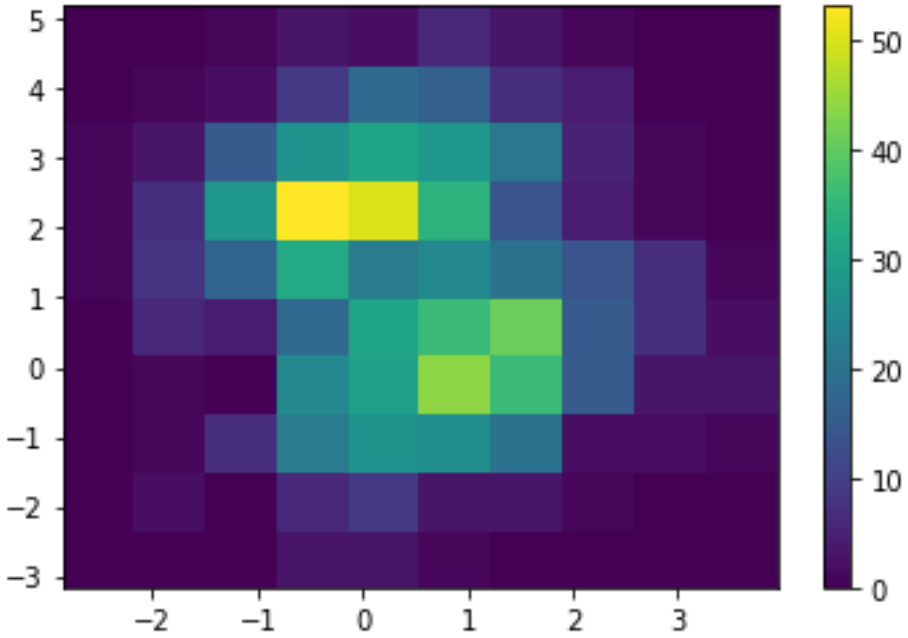
The domain of x-axis = [-10.585185801927734, 17.85034464435909]
The bandwidth = 5



The domain of x-axis = [-20.585185801927736, 27.85034464435909]
The bandwidth = 10

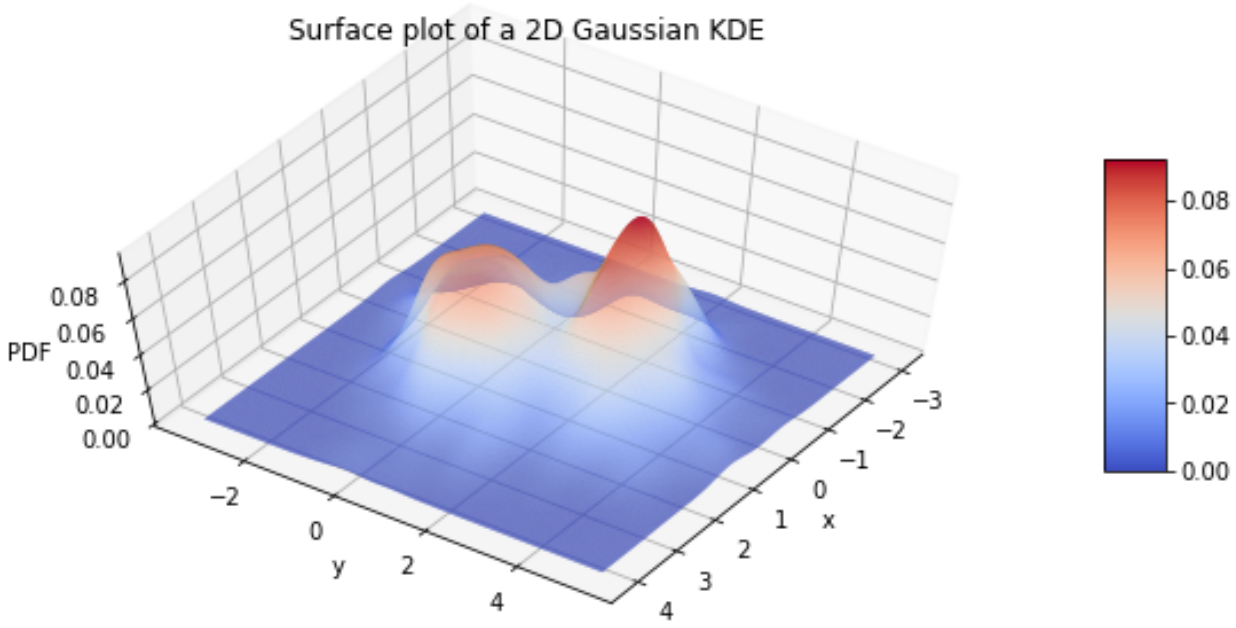


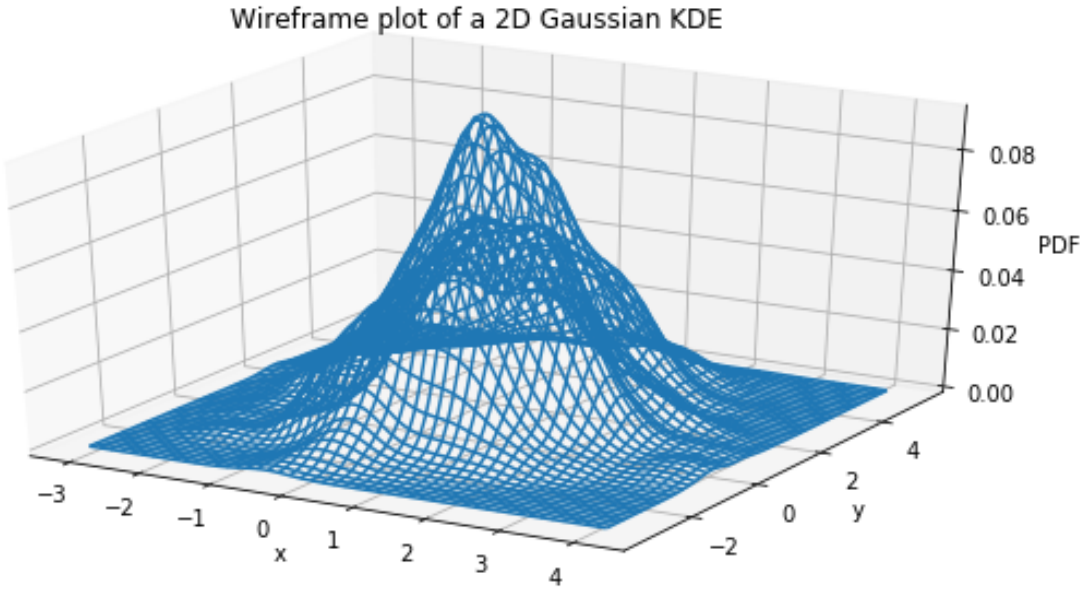
Problem 1.2



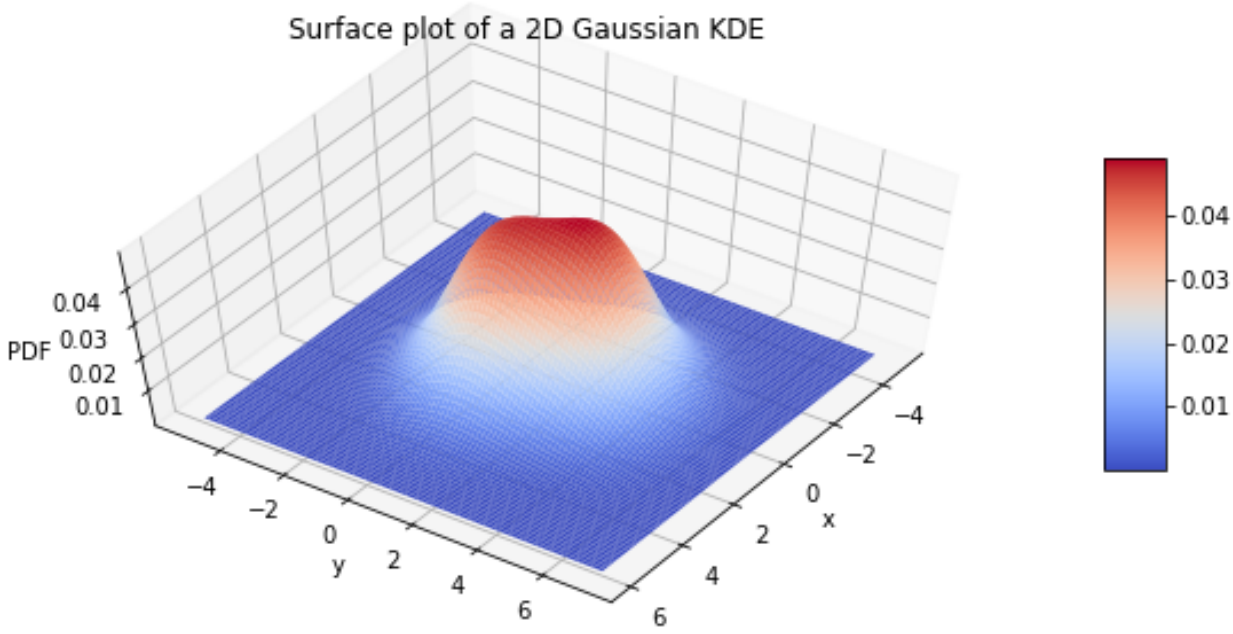
IPSA MISHRA
1001759616

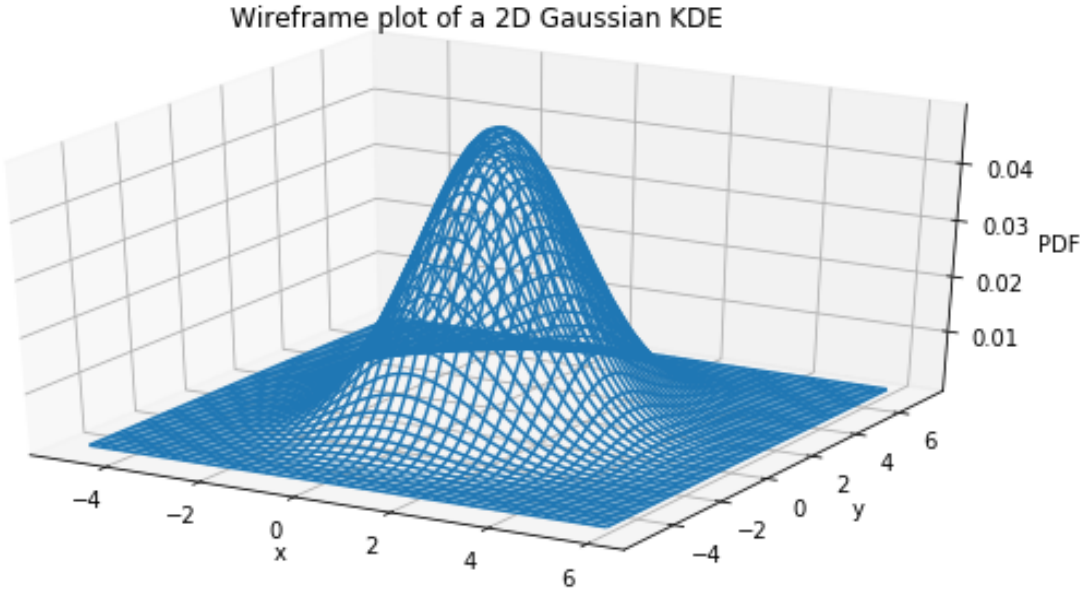
The bandwidth = 0.1
The domain of x-axis = [-3.0483257061090194, 4.148851425509042]
The domain of y-axis = [-3.3592207863752193, 5.38799887039534]



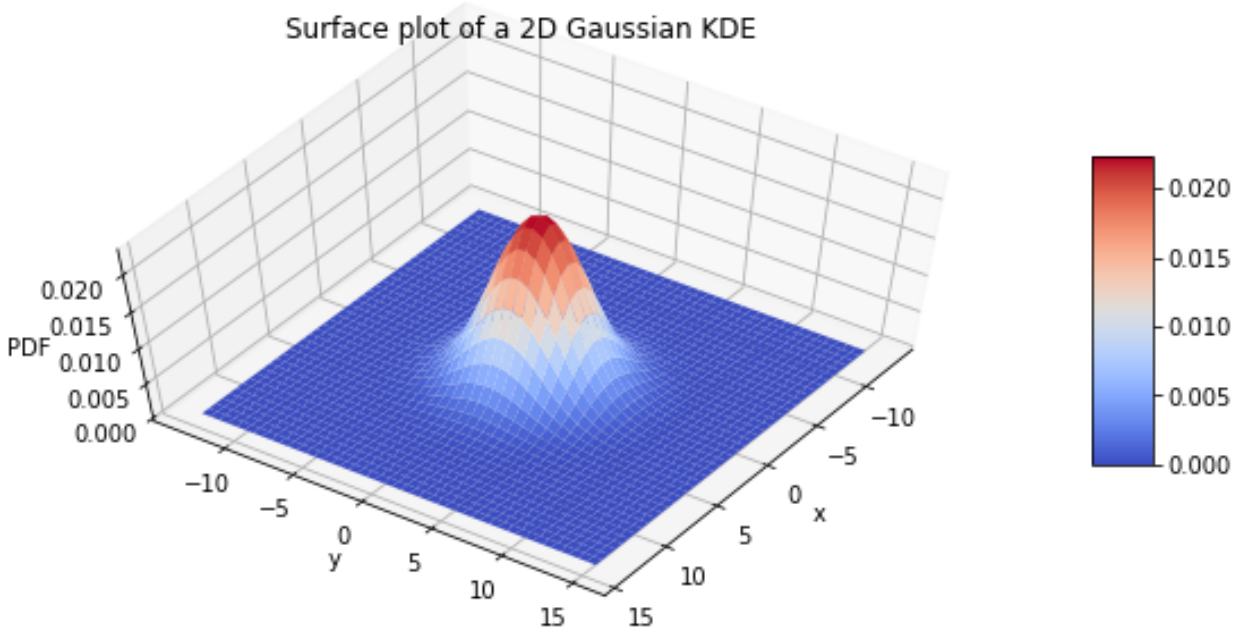


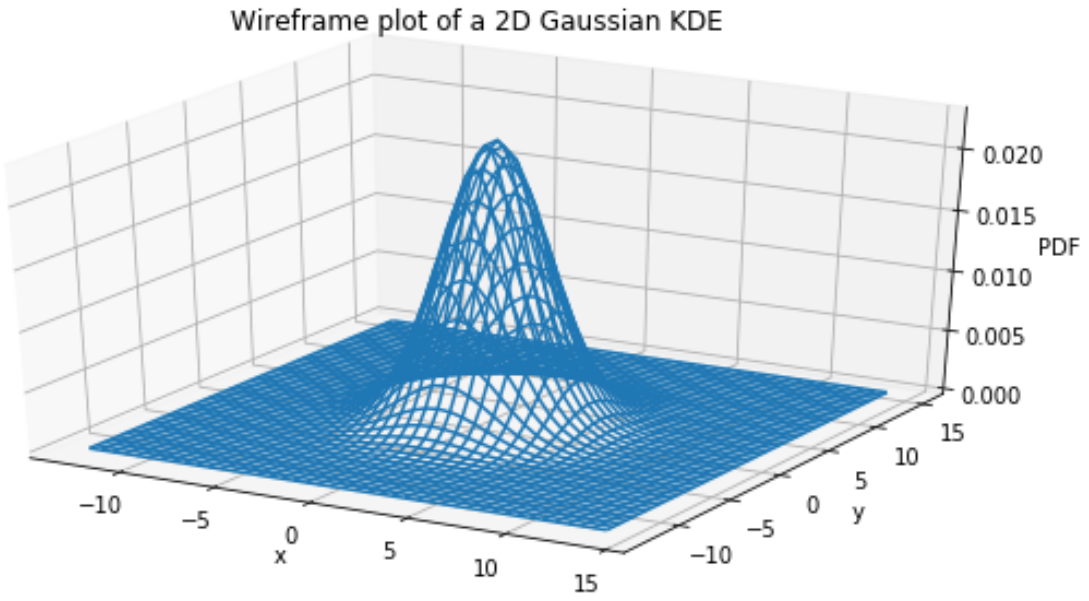
The bandwidth = 1
The domain of x-axis = [-4.848325706109019, 5.948851425509042]
The domain of y-axis = [-5.159220786375219, 7.18799887039534]



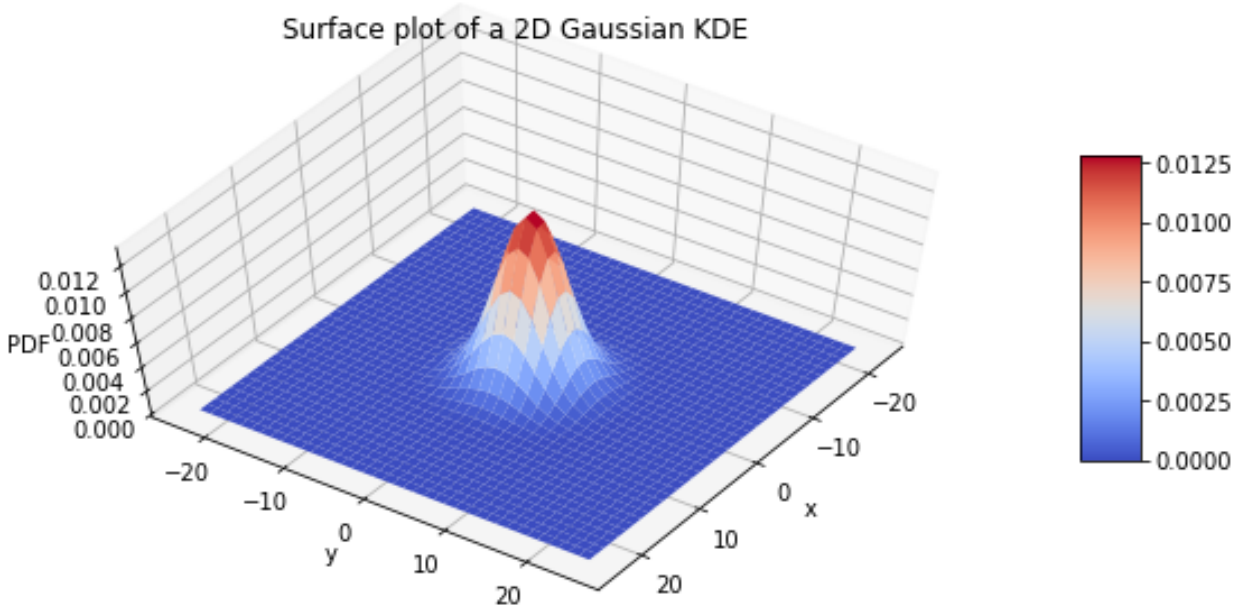


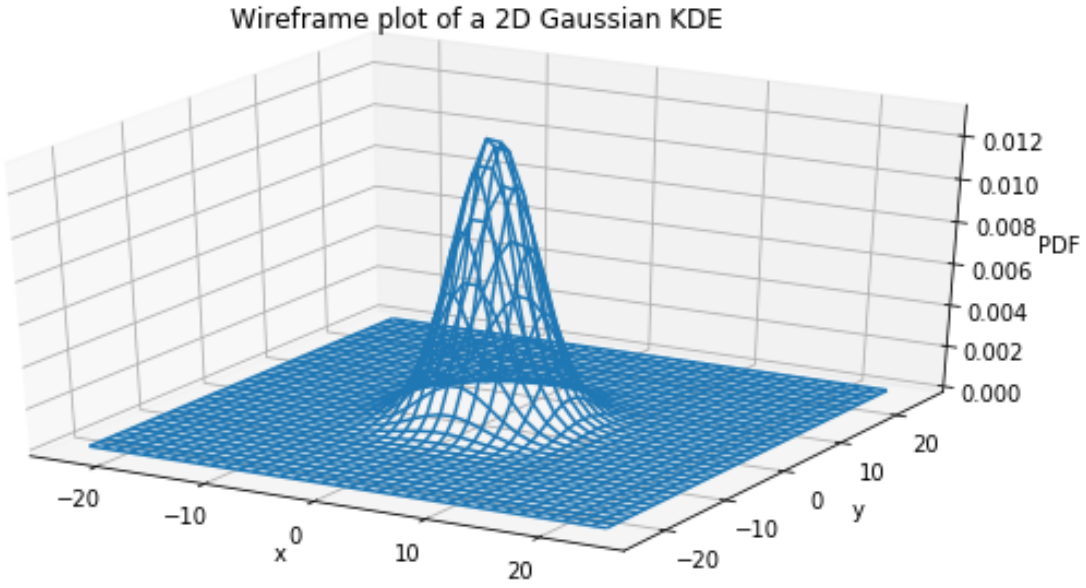
The bandwidth = 5
The domain of x-axis = [-12.84832570610902, 13.94885142550904]
The domain of y-axis = [-13.159220786375219, 15.18799887039534]





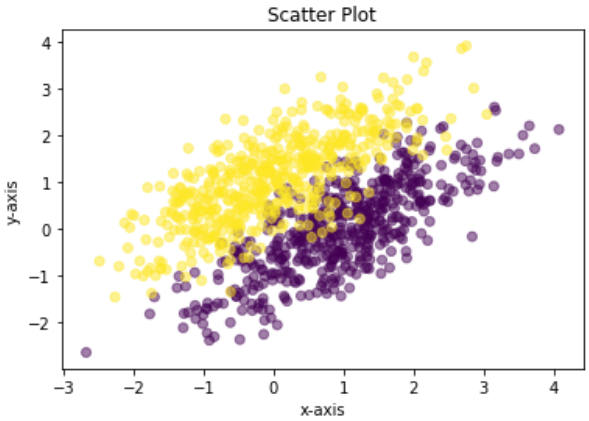
The bandwidth = 10
The domain of x-axis = [-22.84832570610902, 23.94885142550904]
The domain of y-axis = [-23.15922078637522, 25.18799887039534]





Problem 2.1 & 2.2

Scatter plot for the complete training data



Metrics for dataset when 500 data points in each class are used. . Averaged over 10 experiments.

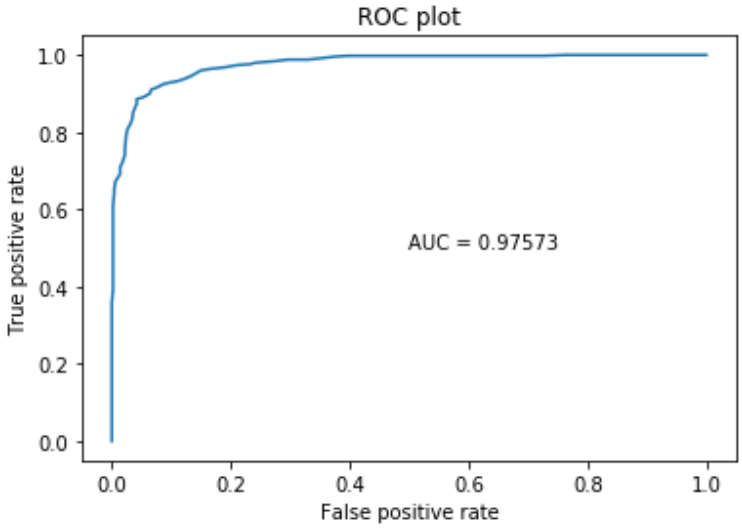
Average Accuracy: 0.9218999999999999

Average AUC: 0.9766505999999999

Average Precision: 0.9219152130329824

Average Recall: 0.9222000000000001

Average of Confusion Matrices: [[461.1 38.9]
[39.2 460.8]]



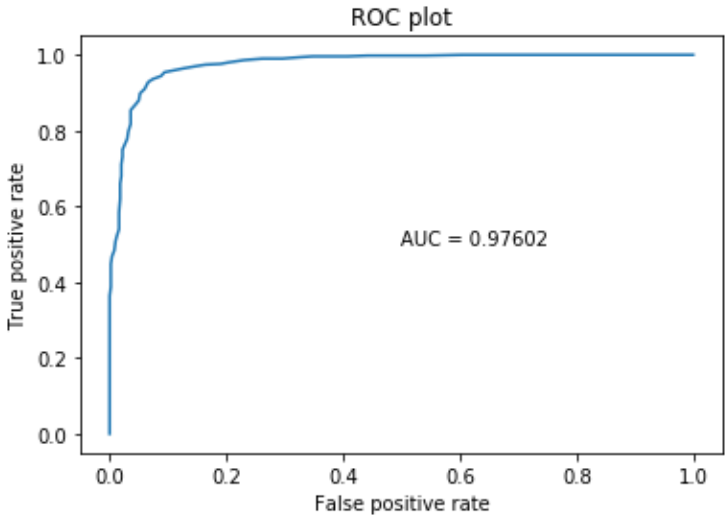
Metrics when 700 data points in class 0 and 300 data points in class 1 of training dataset is used. Averaged over 10 experiments.

Average Accuracy: 0.785

Average AUC: 0.9758954

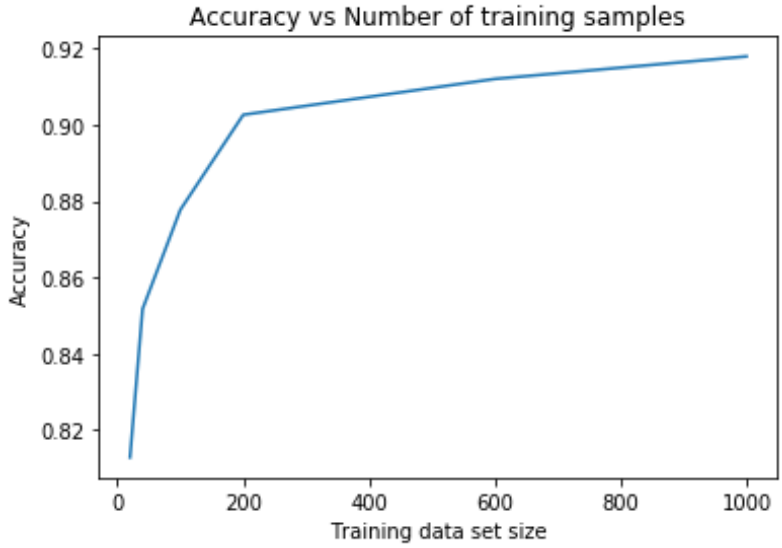
Average Precision: 0.9906239021632804

Average Recall: 0.5753999999999999
Average of Confusion Matrices: [[287.7 212.3]
[2.7 497.3]]



We can see that when the dataset is unbalanced the accuracy reduces. The reason is that the Naive bayes classifier also uses a high prior for class 0 and hence in test set lot of the data points belonging to class 1 get wrongly labeled as class 0 which affects the accuracy.

Varying the class size according to [10; 20; 50; 100; 300; 500]



We can see that the accuracy improves as the training dataset size increases. This is because the naive bayes classifier is unable to accurately determine the mean and covariance of the underlying distribution when we have fewer samples.

IPSA MISHRA
1001759616

Problem 2.3

5-fold cross-validation was done with the 'Amazon Reviews' dataset by using 2D tf-idf vectors generated in previous assignment. The reason for using lower dimension is that we have a total of 199 reviews and having larger dimensional features would affect the quality of the naive Bayes classifier.

Average Accuracy: 0.6687179487179488
Average Precision: 0.6346627299568477
Average Recall: 0.9742424242424244