INDIAN INSTITUTE OF TECHNOLOGY GOA

CS331

[Machine Learning]

Lab Assignment 1

Submitted By Group :-

OmNiSiYaKrMa

1). Create a synthetic dataset for the credit default problem. You may choose priors suitably. Generate 10^4 points from the model. Using the generated dataset evaluate a) bayes classifier and b) classifier which has a randomization built in.

For the synthetic dataset, Prior was set to:

$p_0 = 0.8$ *(non-defaulter)*

$p_1 = 0.2$ *(defaulter)*

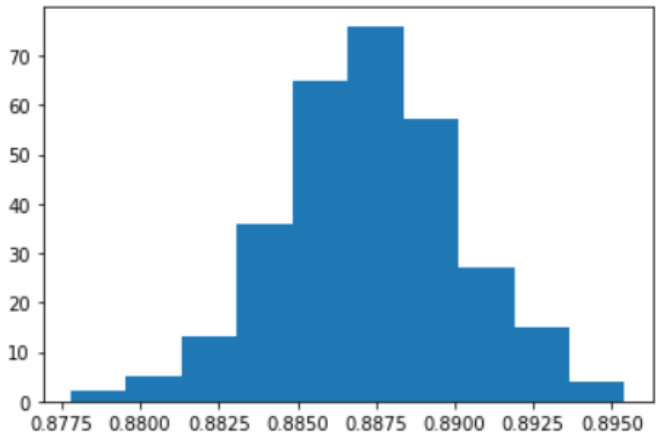
For credit default problem, we generate feature data from following normal distributions:

1) For label $y = 0, \boldsymbol{\mu}_0 = -1, \boldsymbol{\sigma}_0 = 1$
2) For label $y = 1, \boldsymbol{\mu}_1 = 1, \boldsymbol{\sigma}_1 = 1$

The normal distributions: $N \sim (-1, 1)$ and $N \sim (1, 1)$ respectively for labels $y = 0, y = 1$.

After simulating these 10^4 sample points for 300 times, we got the following distribution for Bayes Classifier:

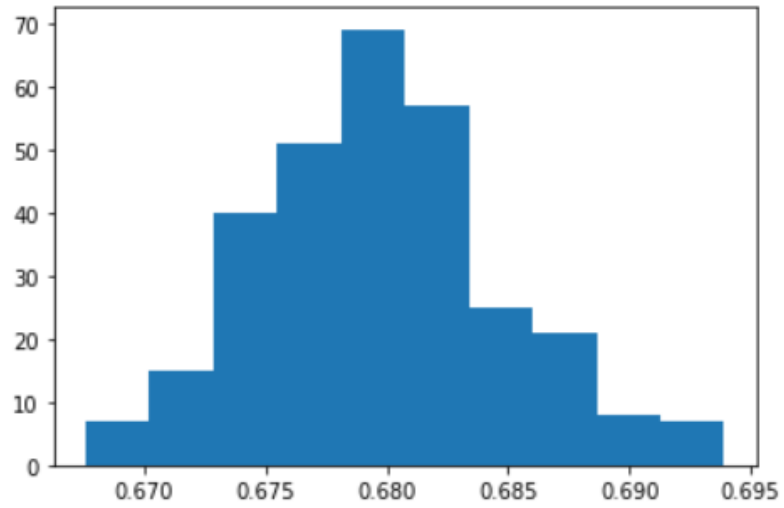
```
import matplotlib.pyplot as plt
plt.hist(acc_list,bins=10)
plt.show()
```



Mean accuracy for 300 simulations is close to ***0.8875***.

Next we used classifier with randomizer for evaluation of the same:

```
:  import matplotlib.pyplot as plt
   plt.hist(acc_list,bins=10)
   plt.show()
```

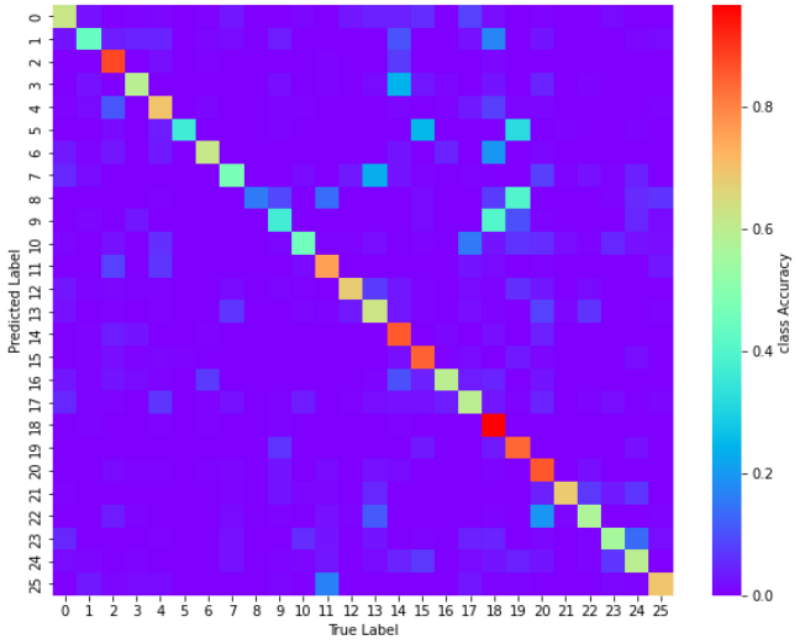


Here, for the same data mean accuracy is quite low i.e. close to ***0.680***. So, we conclude that a randomizer classifier is less efficient than Bayes classifier.

2). **On the Handwritten dataset attached, evaluate the bayes classifier with naive bayes assumption. You may rescale the pixel values to 0,1. As described in class, split the dataset randomly to 80:20 to training and test dataset.**

For this code, we used sklearn library to implement Naive Bayes classifier. We initially normalized the values for all rows in columns 1 to 784 to {0,1} and divided the dataset in 80:20 ratio as given for training and testing.

After applying GaussianNB(), we found out that the accuracy was around ***50%***, the reason behind not being an optimal classifier can be that it assumes normal distribution for feature data whereas our feature data is discrete. We also tested MultinomialNB() which gave an accuracy of about ***70%*** which is fairly high as it is based on multinomial distribution(similarly, for Bernoulli as it is based on discrete RV).

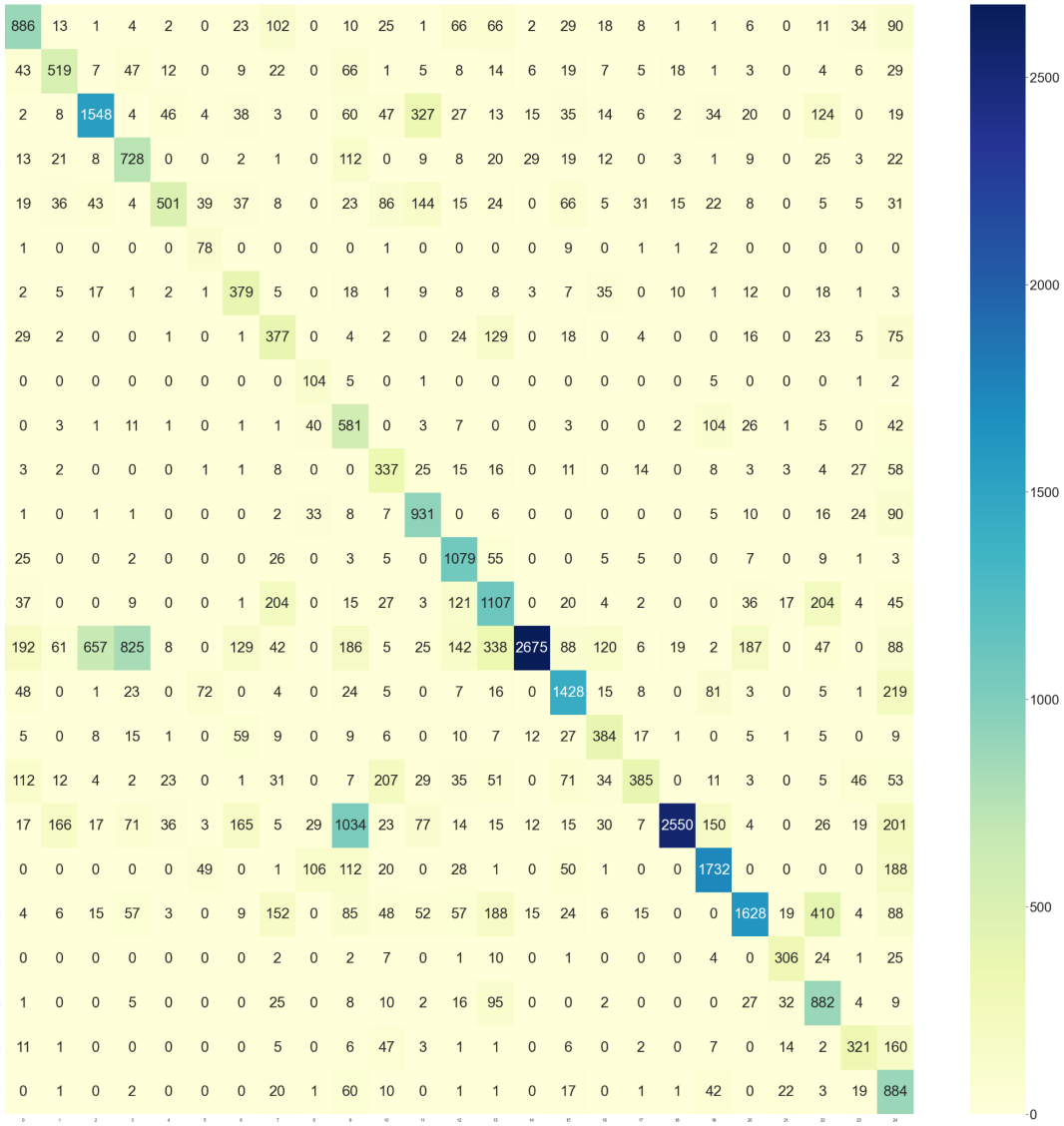


Confusion Matrix(using MultinomialNB() classifier) for all classes(0 to 25)

One of the key observations in the dataset is that class 14 & 18 i.e. letter 'O' & 'S' respectively have the most images thus, while testing we can observe in the above confusion matrix that their prediction has high accuracy.

3). **Repeat 2 by implementing your own naive bayes classifier.**

In this problem, we implemented our own naive bayes classifier by calculating priors and class conditionals. Posteriors were calculated for all classes (0 to 25) and then the label with maximum posterior was selected. For ease and accuracy of calculation we maximized log(qi's) = log(pi's)+log(fi's) for all classes. The accuracy of this program was also around ***70%*** (similar to MultinomialNB()).



Confusion Matrix(using our own naive bayes classifier) for all classes(0 to 25)

References:

- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bernoulli.html