Assignment 1 Report

Sonu Dixit

I have used PYTHON 3.

Task-1: Implementation of Random Projection

- Code File : projections.py
- A random matrix of required size is created by sampling its elements from Gaussian distribution with mean=0 and std = 1/finalDimensionSize. Input matrix is multiplied with this random matrix for dimensionality reduction.

Task-2: Bayes and NN classifier implementation

- Code File : BayesClf.py, Nearest_Neighbour.py
- For Bayes Classifier, Wherever data is continuous(Dolphins and Pubmed dataset) Normal distribution has been assumed. By using ML estimate mean(mean of samples) and standard deviation(std of samples) of features has been calculated. For Twitter dataset data has been considered as categorical. I have assumed Multinomial distribution for calculating ML parameters. All features for all the three datasets has been considered independent.
- For NN classifier, I have treated all the three datasets equally.

Task-3: Comparision of Result for low and high dimensional data

Dolphins Dataset : On Full dataset both the bayes and NN classifier give 100% accuracy

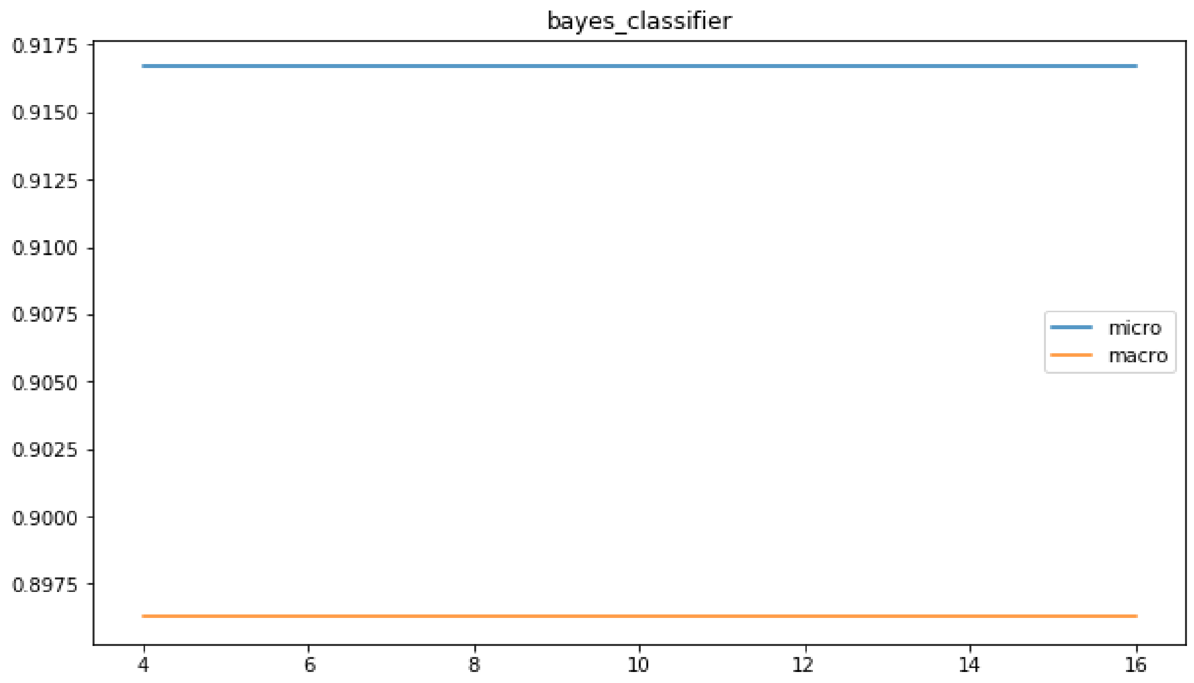


Figure 1: Because of decrease in data dimesion, performance of Bayes has decreased. But it has remained constant from 4 dimesion to 16 dimension.

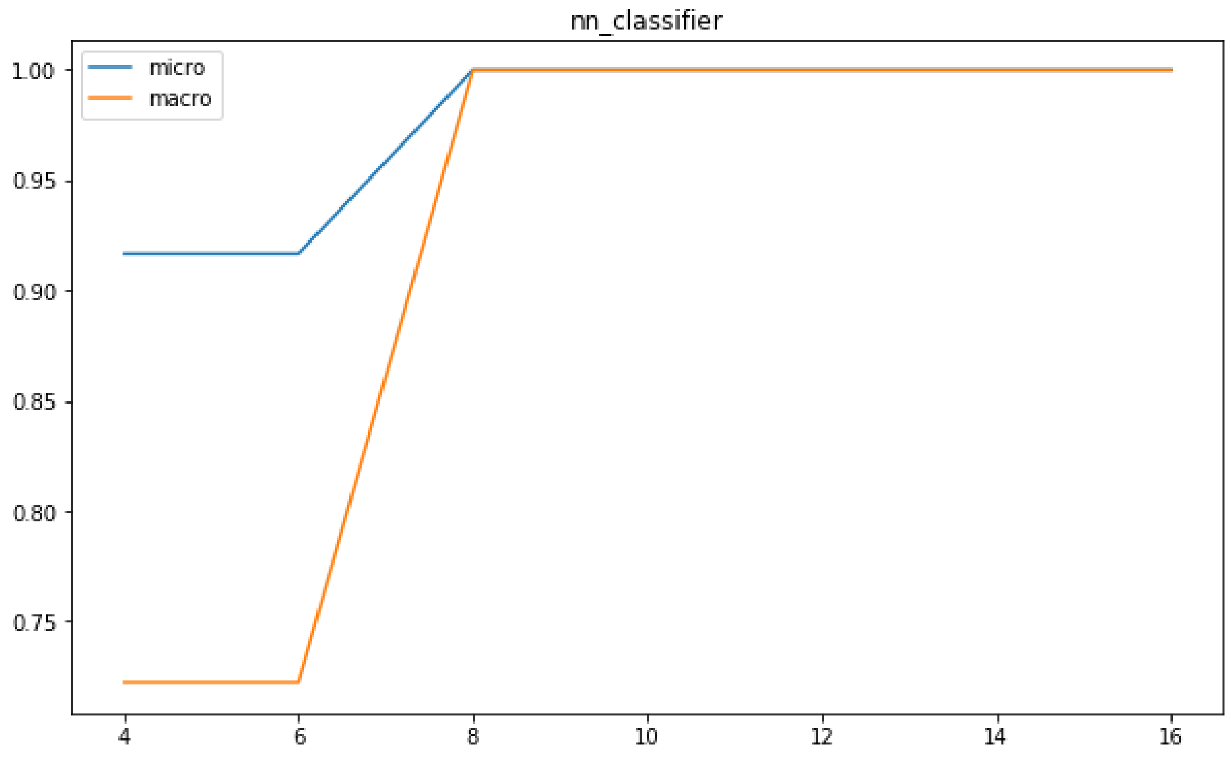


Figure 2 Initially accuracy is poor, but with 8dimensional data, it achieves 100% accuracy.

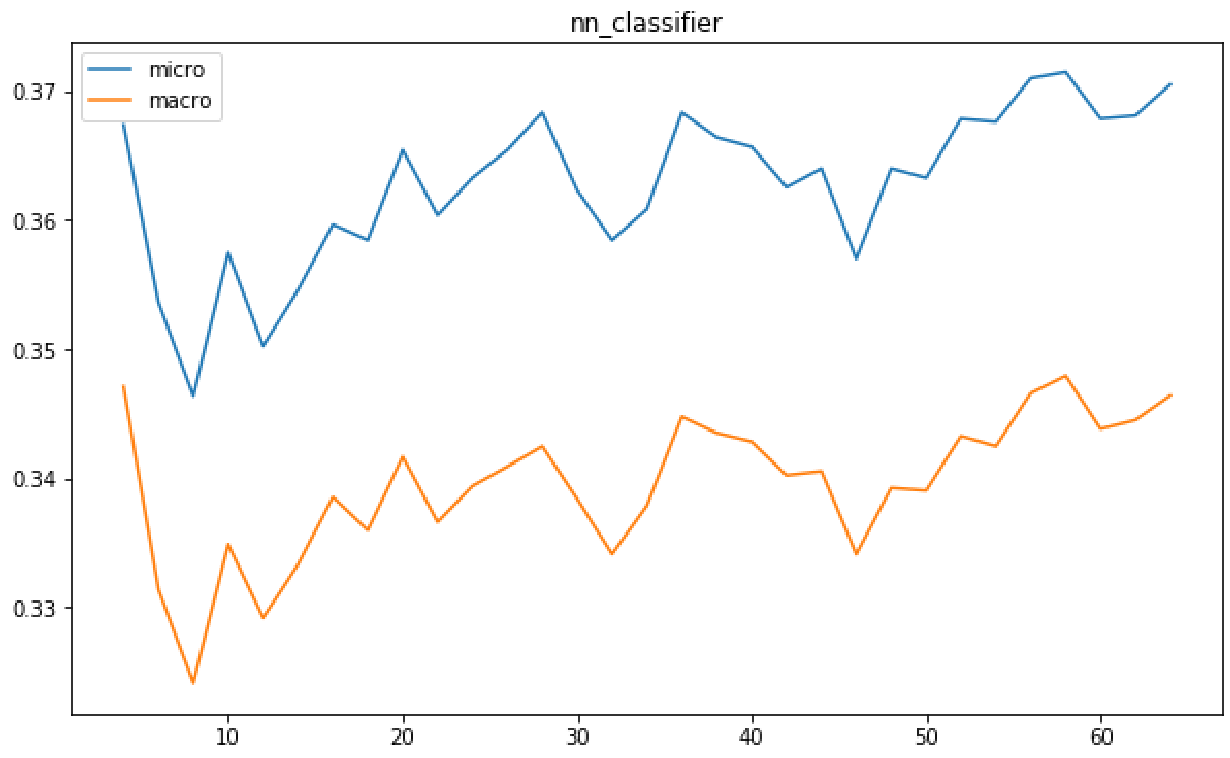
pubmed dataset full dimension

NN:

Test Accuracy :: 37.46

Test Macro F1-score :: 34.88

Test Micro F1-score :: 37.46



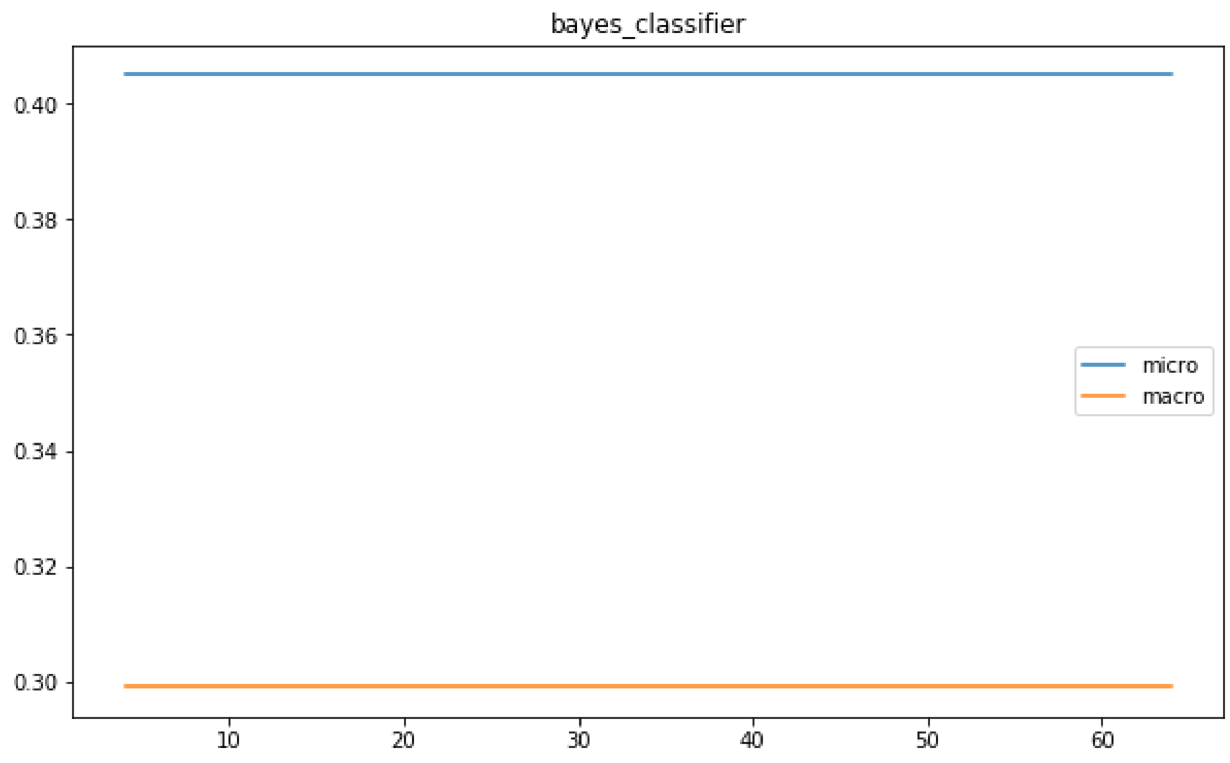
NN classifier gave its best performance at least dimension(4). After that performance has reduced. But overall as the dimension is increasing, performance is also getting better.

Bayes:

Test Accuracy :: 42.99

Test Macro F1-score :: 31.88

Test Micro F1-score :: 42.99



Even with the reduced dimension, Bayes classifier is giving good accuracy.

Twitter Dataset full dimension:

These preprocessing steps are same for all tasks.I have considered bag of words model. Rare words(length 2,3,4 and frequency <5) has been removed from vocab. Laplace smoothening has been used for avoiding zero probability.

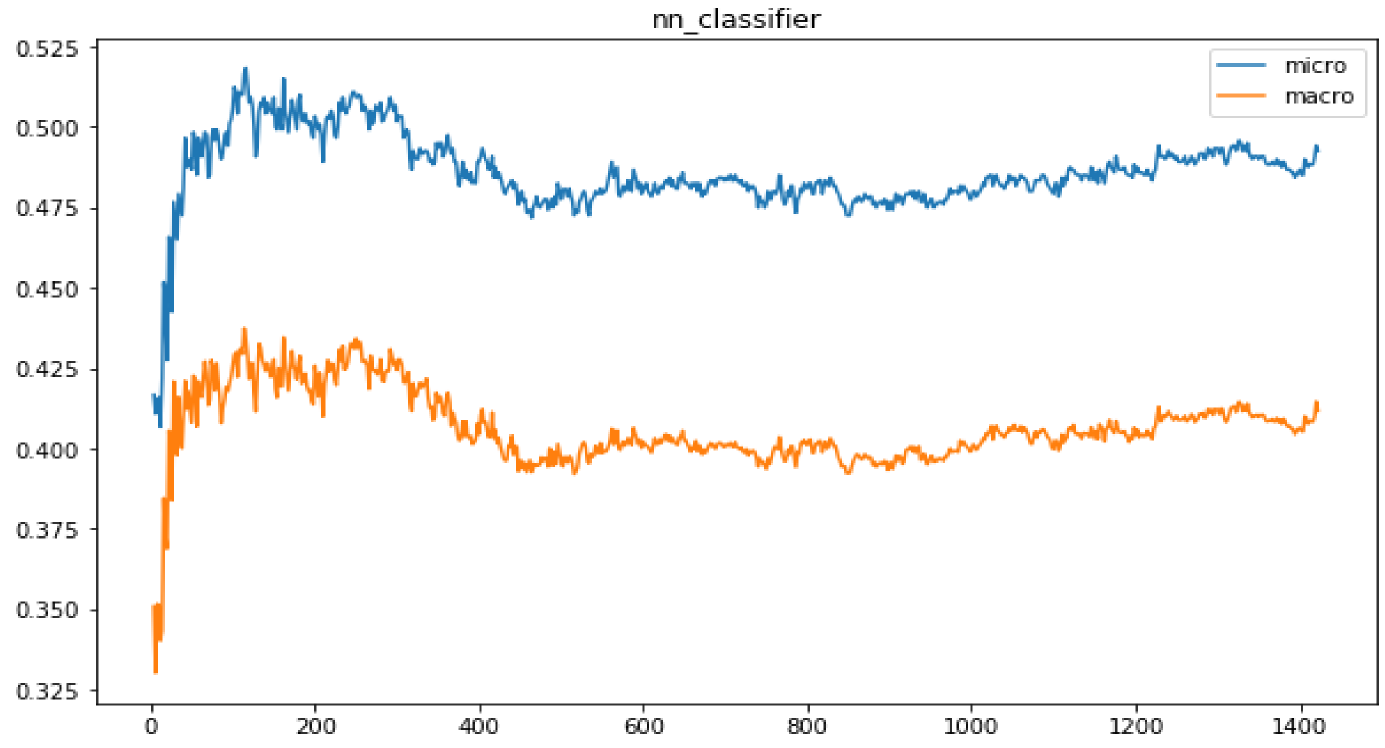
Nearest Neighbour

Test Accuracy :: 49.00

Test Macro F1-score :: 40.57

Test Micro F1-score :: 49.00

We get the best accuracy at lower dimension than the full dimension. as it can be seen in the graph the best accuracy as around 125 dimension this is this might be because the data Matrix for Twitter dataset will be very sparse. So in this case reducing the dimensionality helps improve the performance.

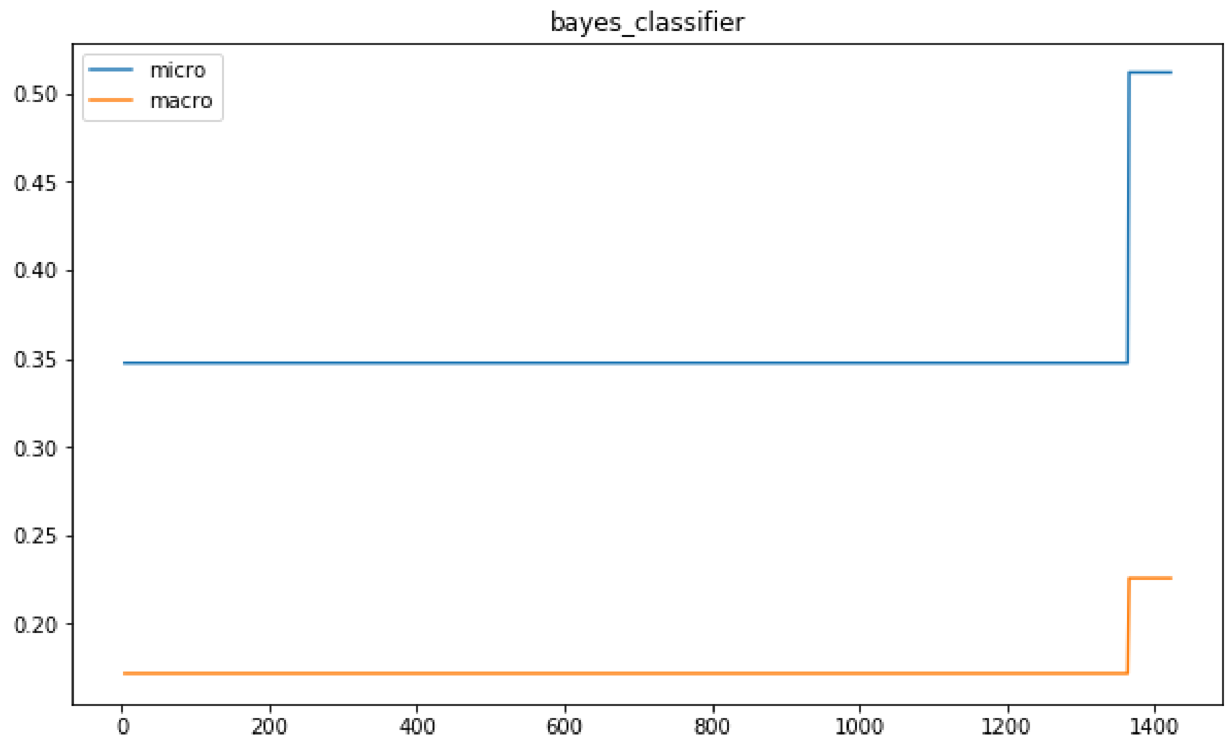


Bayes Classifier :

Test Accuracy :: 57.66

Test Macro F1-score :: 50.78

Test Micro F1-score :: 57.66



For bayes classifier as the dimensionality is been increased the accuracy is being increased. from the graph it can be inferred that we can get the best accuracy at lower dimension then the whole dimension which was 2800 in this case.

a point to note is its performance has been almost constant from around 8 dimension to 1300 dimensions. Accuracy got sudden increase at around 1400 dimension.

TASK-4 : USING library functions

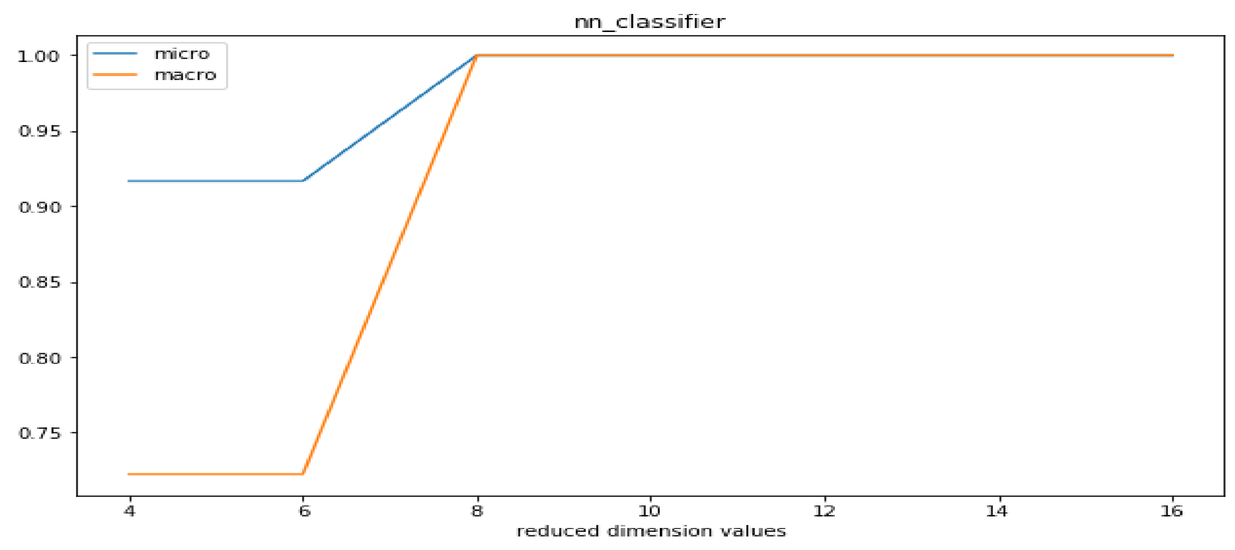
dolphins dataset full dimension

Nearest Neighbour

Test Accuracy :: 100.00

Test Macro F1-score :: 100.00

Test Micro F1-score :: 100.00

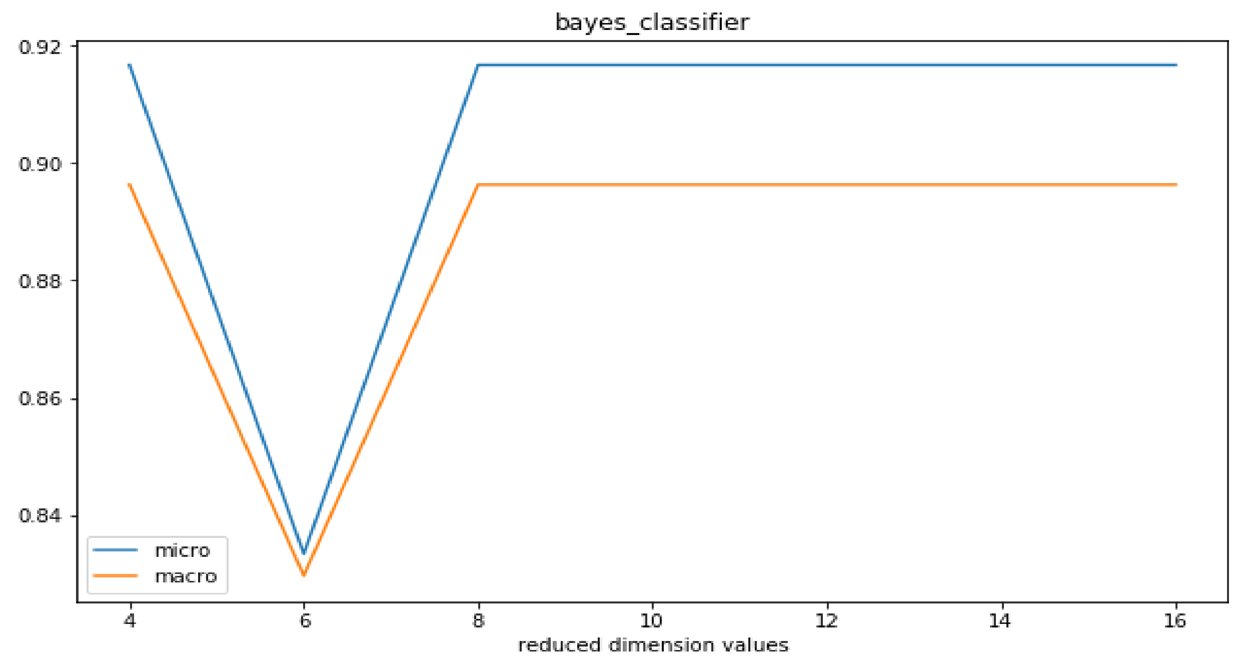


Bayes Classifier

Test Accuracy :: 100.00

Test Macro F1-score :: 100.00

Test Micro F1-score :: 100.00



pubmed dataset full dimension

Nearest Neighbour

Test Accuracy :: 37.46

Test Macro F1-score :: 34.88

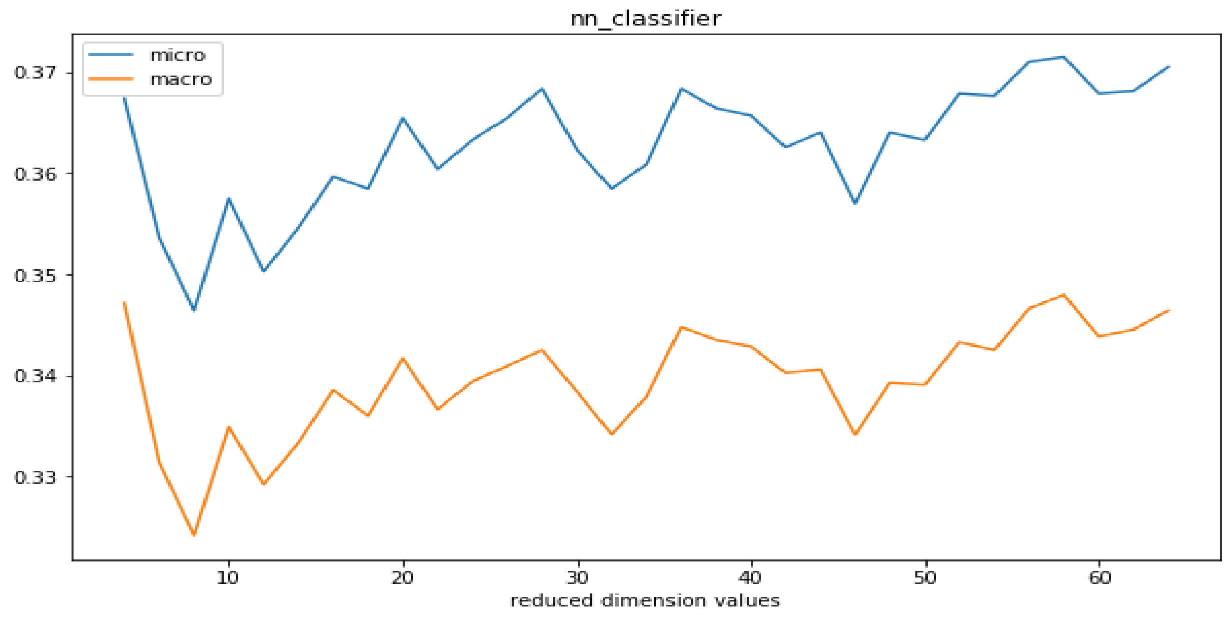
Test Micro F1-score :: 37.46

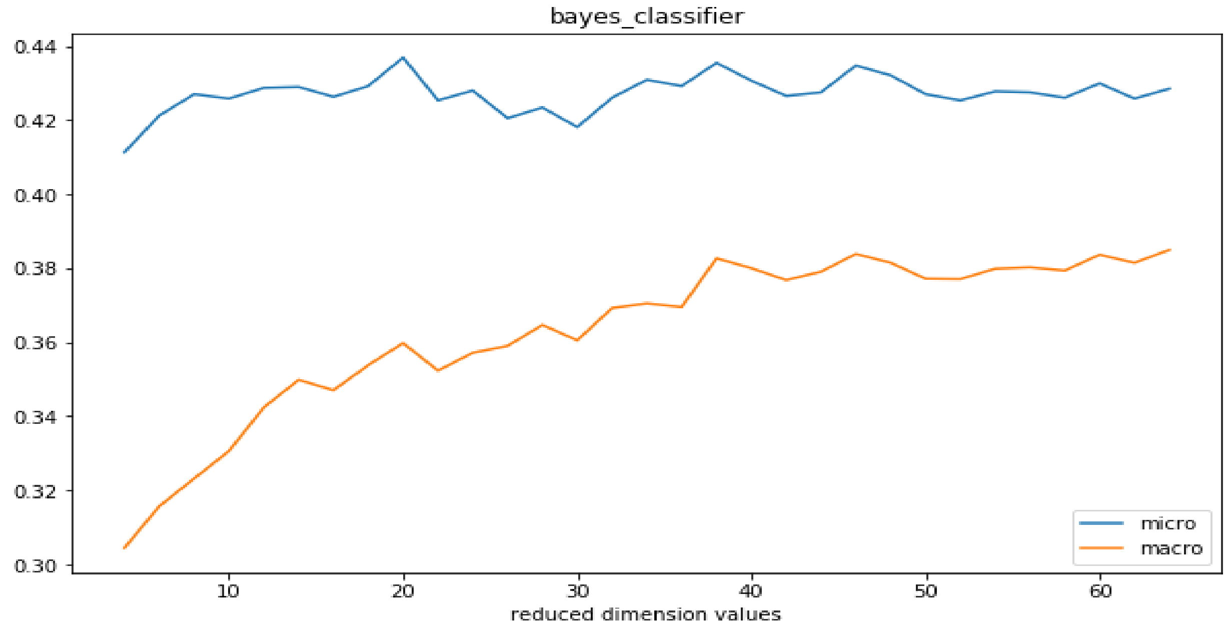
Bayes Classifier

Test Accuracy :: 42.29

Test Macro F1-score :: 39.40

Test Micro F1-score :: 42.29





twitter dataset full dimension

Nearest Neighbour

Test Accuracy :: 48.91

Test Macro F1-score :: 40.72

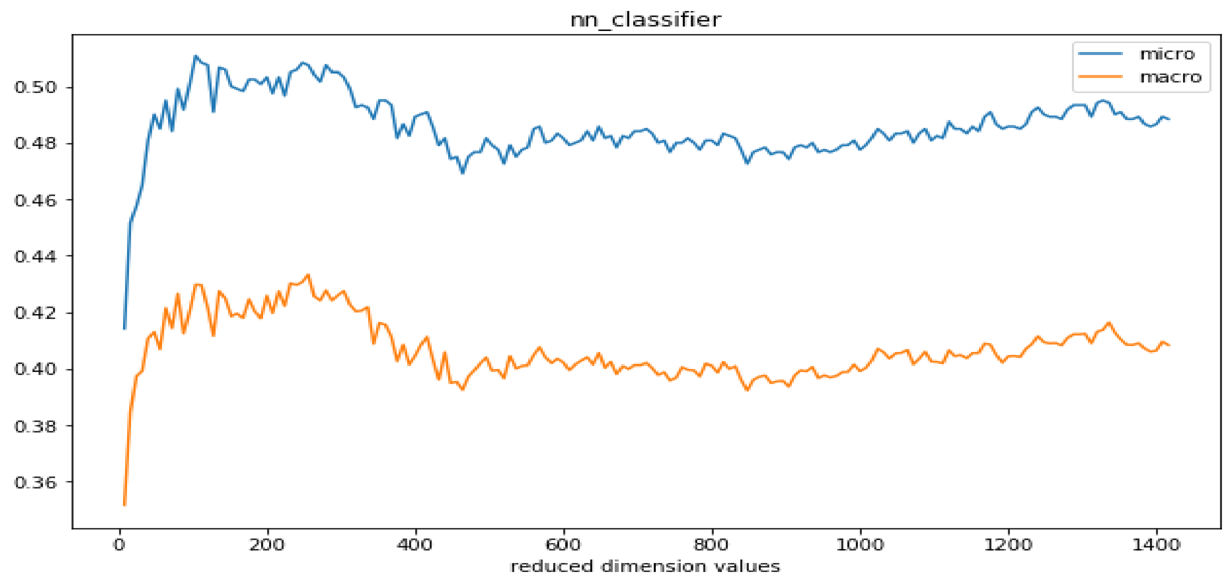
Test Micro F1-score :: 48.91

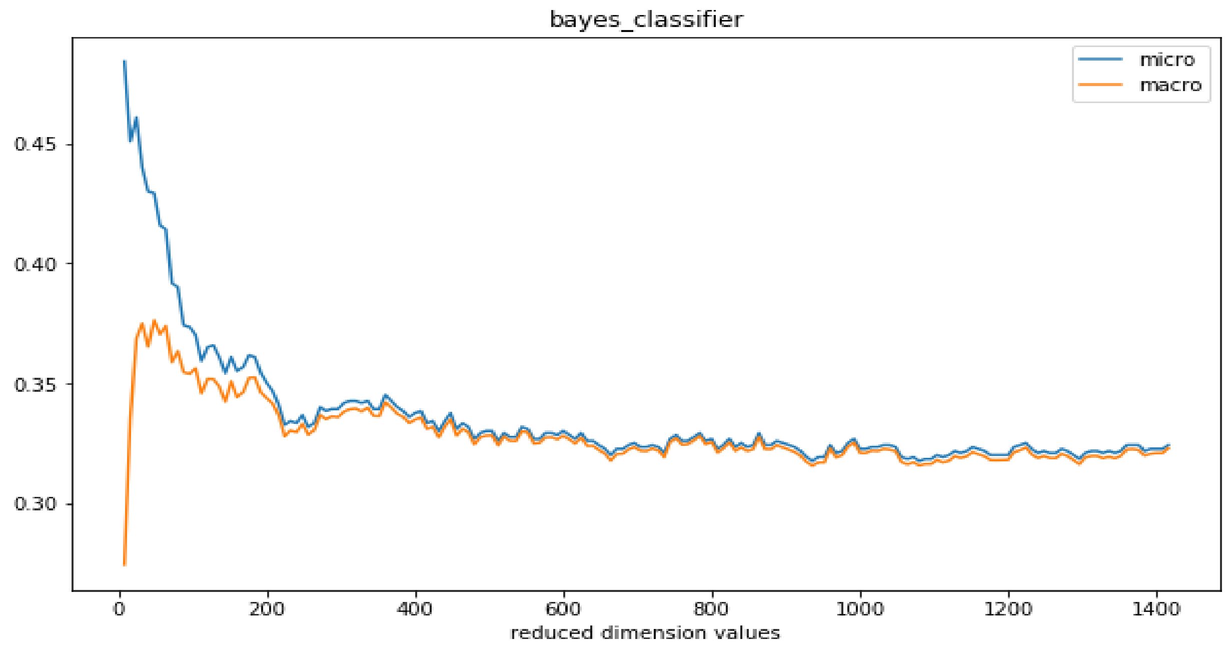
Bayes Classifier

Test Accuracy :: 57.66

Test Macro F1-score :: 50.78

Test Micro F1-score :: 57.66





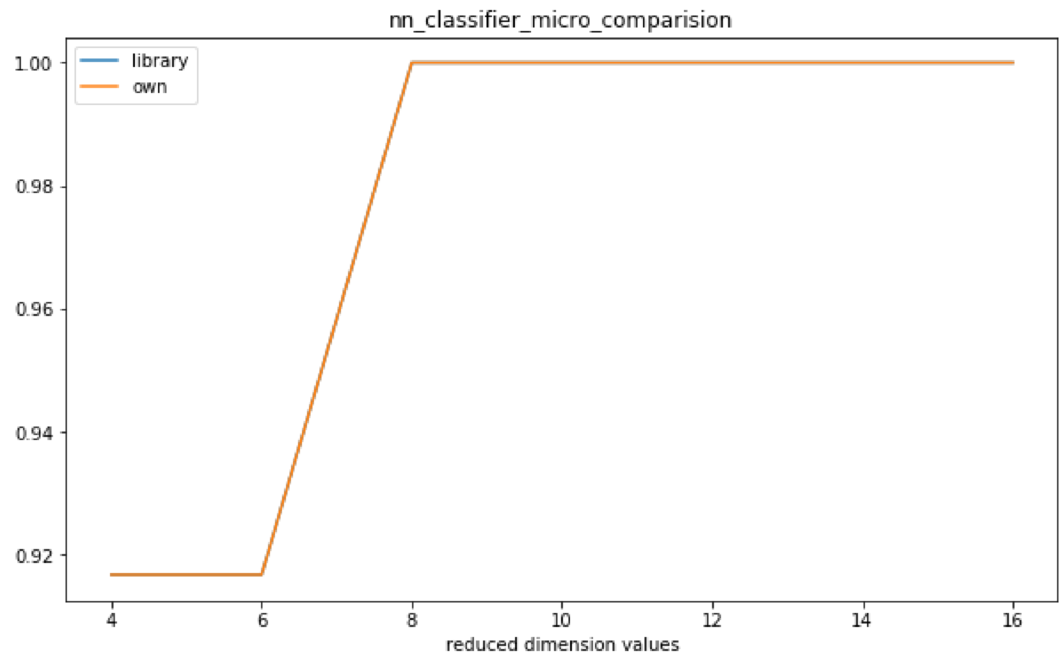
TASK-5 Comparision of Scikit library and own implementation

1. **Dolphins dataset:**

   On Full dimension both are giving 100% accuracy.

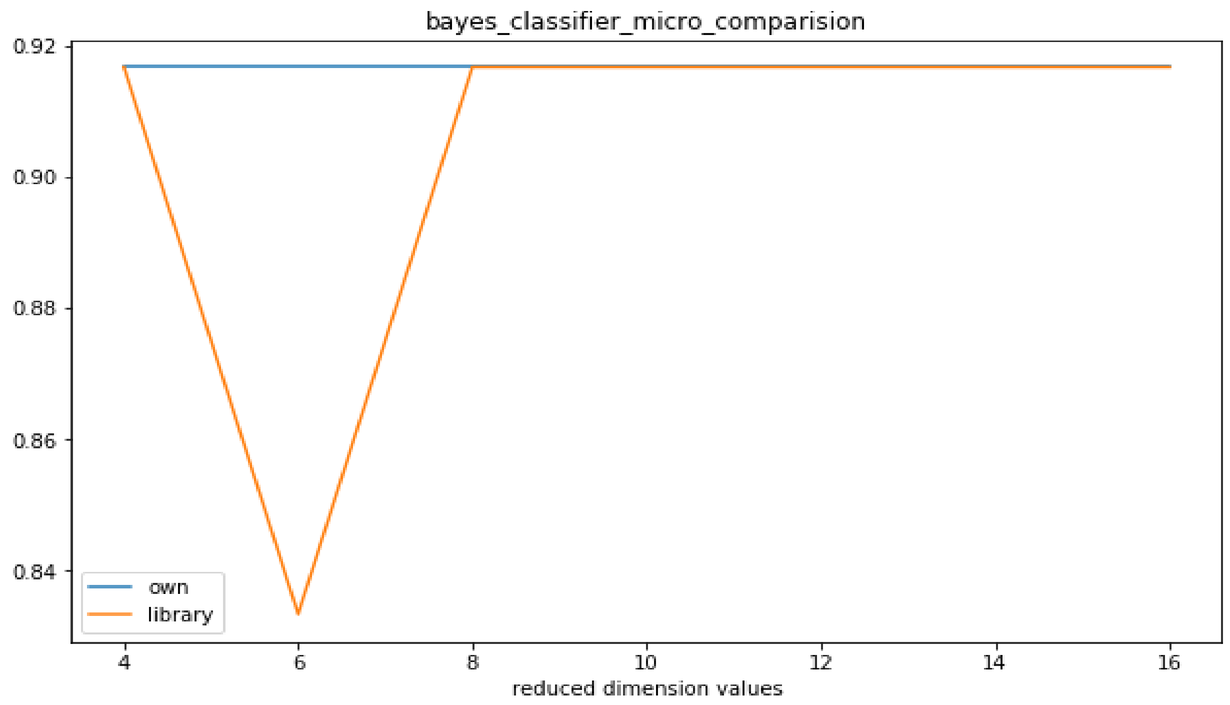
   For reduced dimension following is the plot:

   Nearest Neighbor:



   Both the implementation is equivalent.

   Bayes



Performance of my implementation is better for lower dimension(6). Otherwise the performance is equivalent. With increase in dimension(4->6) performance of library implementation has reduced.

2. **PubMed Dataset**
   Full Dimension:

NN

acc 0.3746376811594203

macro 0.34881362053702913

micro 0.37463768115942037

Bayes

acc 0.42995169082125606

macro 0.3188982090006239

micro 0.42995169082125606

NN

accuracy 0.374

macro 0.3488

micro 0.37463

Bayes

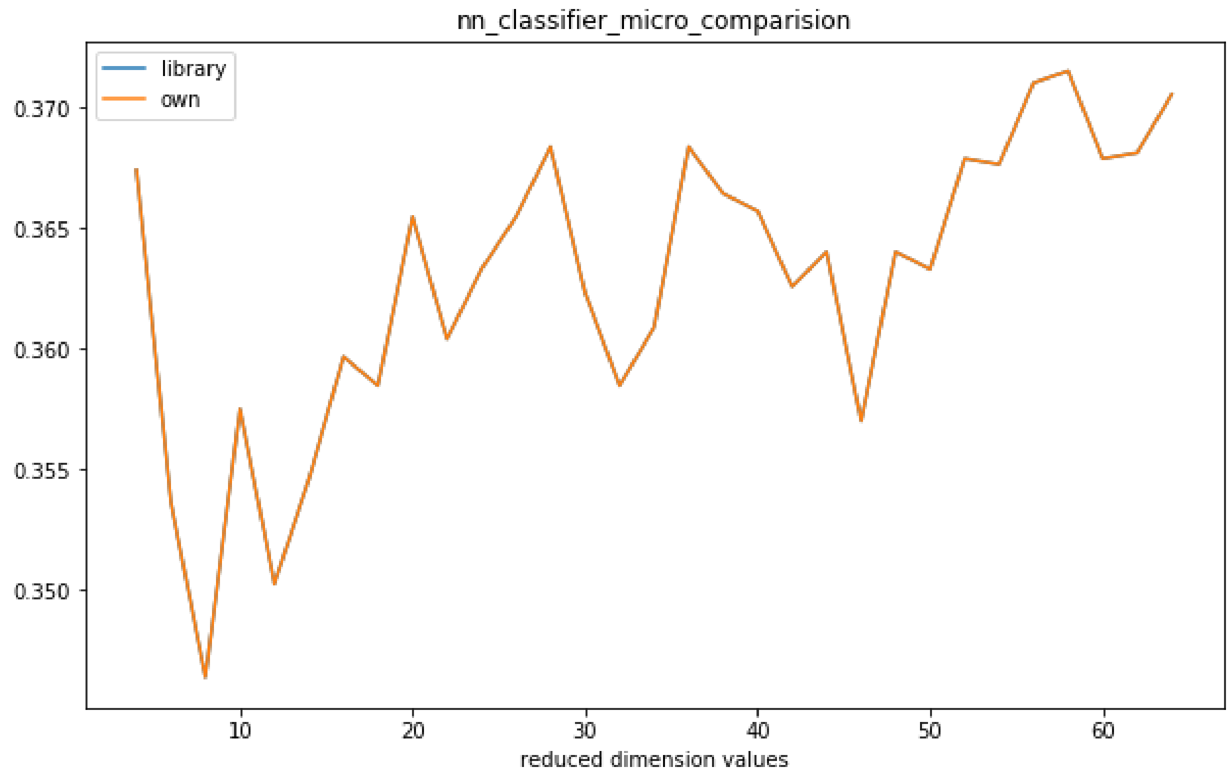
acc 0.42294685990338166

macro 0.39407247578177906

micro 0.42294685990338166

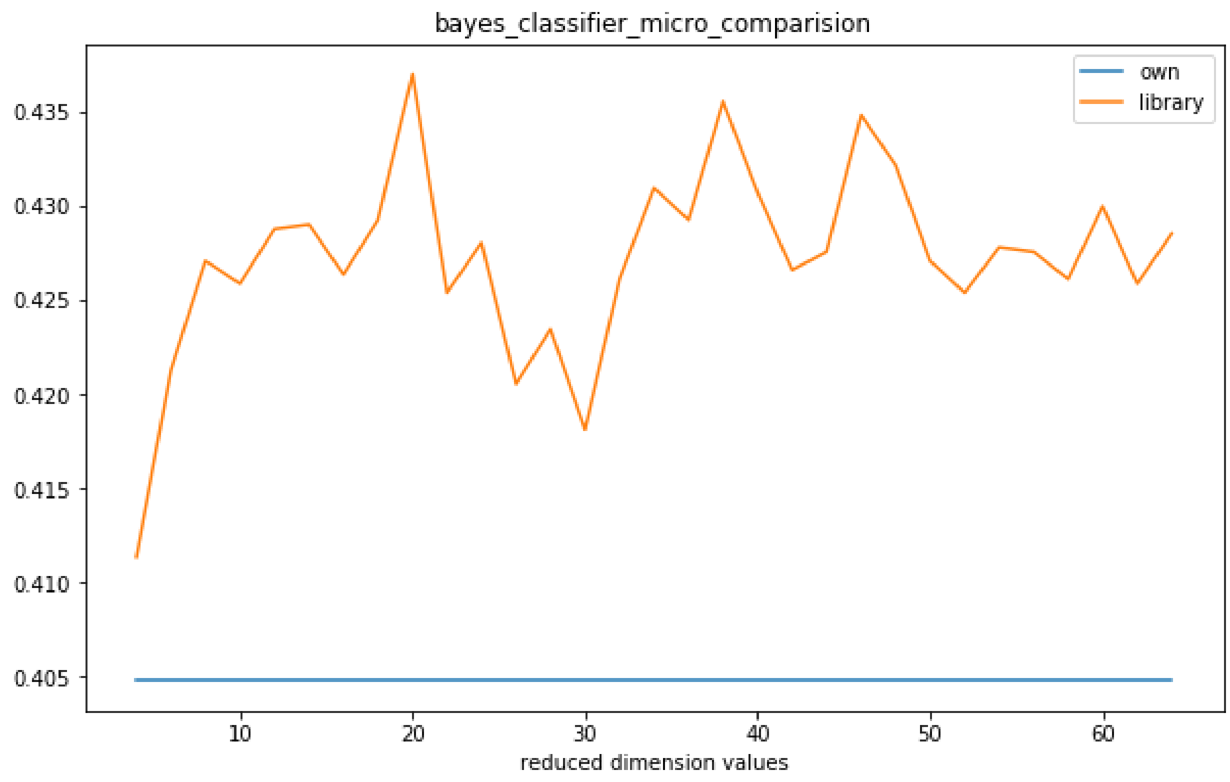
On full dataset both the performance are same.

Comparision for NN on Reduced Dimension:



Both the performance are matching.

Comparision for Bayes on Reduced Dimension:



Library implementation is giving better performance.

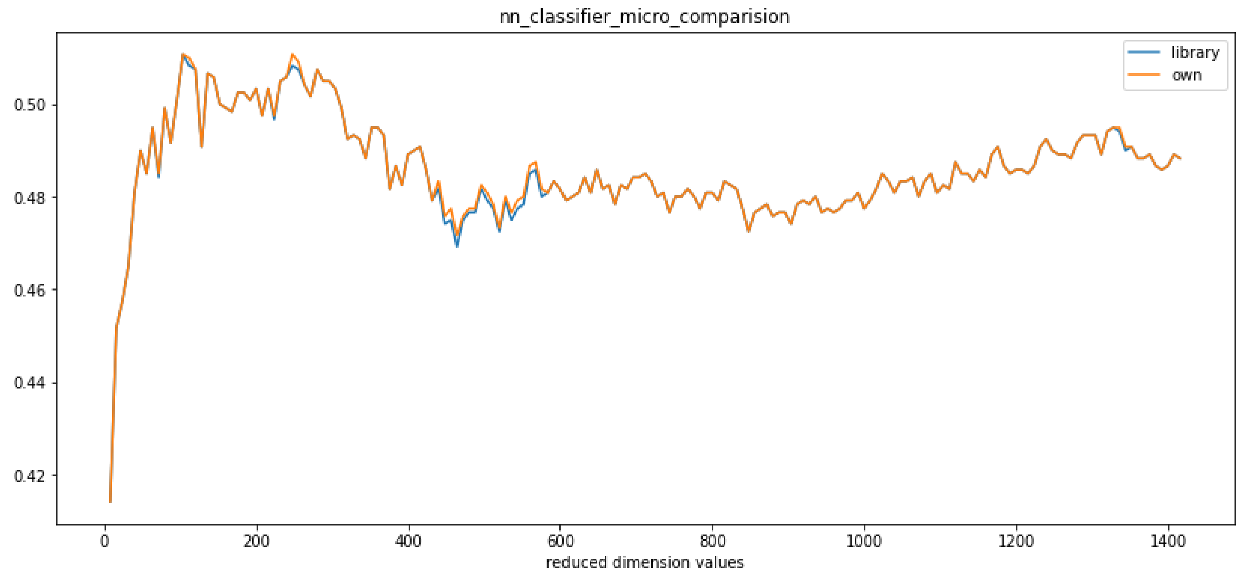
3. **Twitter Dataset**

   Full Dimension

Own Implementation	Library Implementation
NN	NN
acc 0.49	acc 0.4891666666666667
macro 0.40579847181757733	macro 0.4079278112753603
micro 0.49	micro 0.4891666666666667
Bayes	Bayes
acc 0.5766666666666667	acc 0.5766666666666667
macro 0.5078050086114603	macro 0.5078050086114603
micro 0.5766666666666667	micro 0.5766666666666667

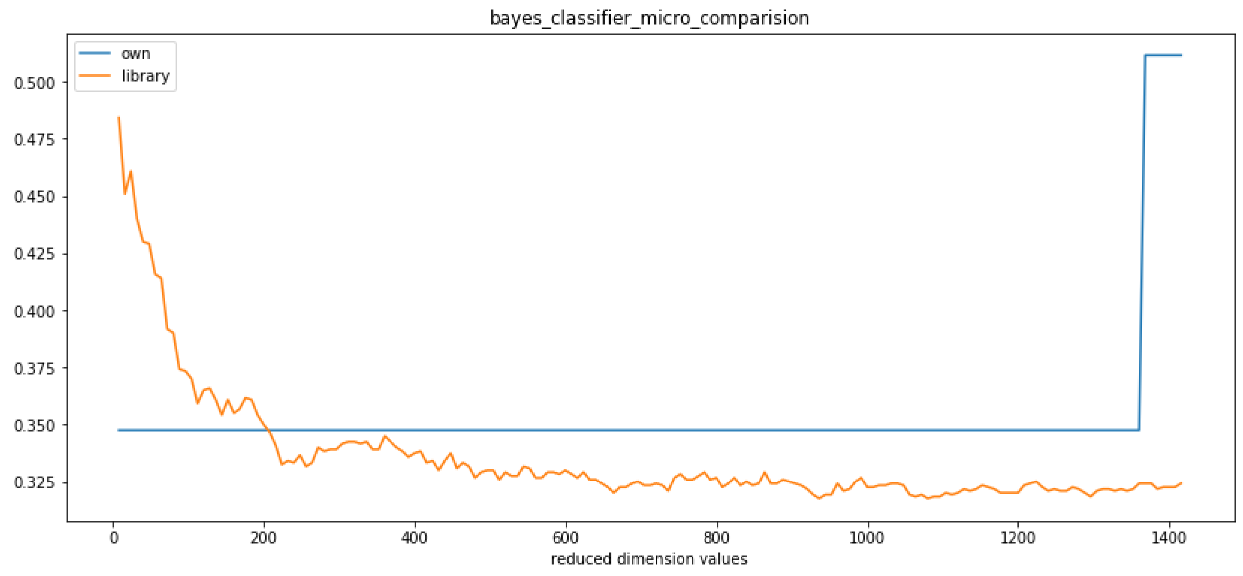
Performance in both cases are equivalent.

Comparision for NN on Reduced Dimension:



Both the implementation are giving same performance. At some simensions library implementation is giving a bit poor results than my implementation.

Comparision for Bayes on Reduced Dimension:



Performance my implementation is almost same, and it got a sudden steep increase at around 1300 dimension. For very low dimension library implementation is giving better performance, with increase in dimension, performance of library implementation is decreasing.

TASK-6 LSH implementation

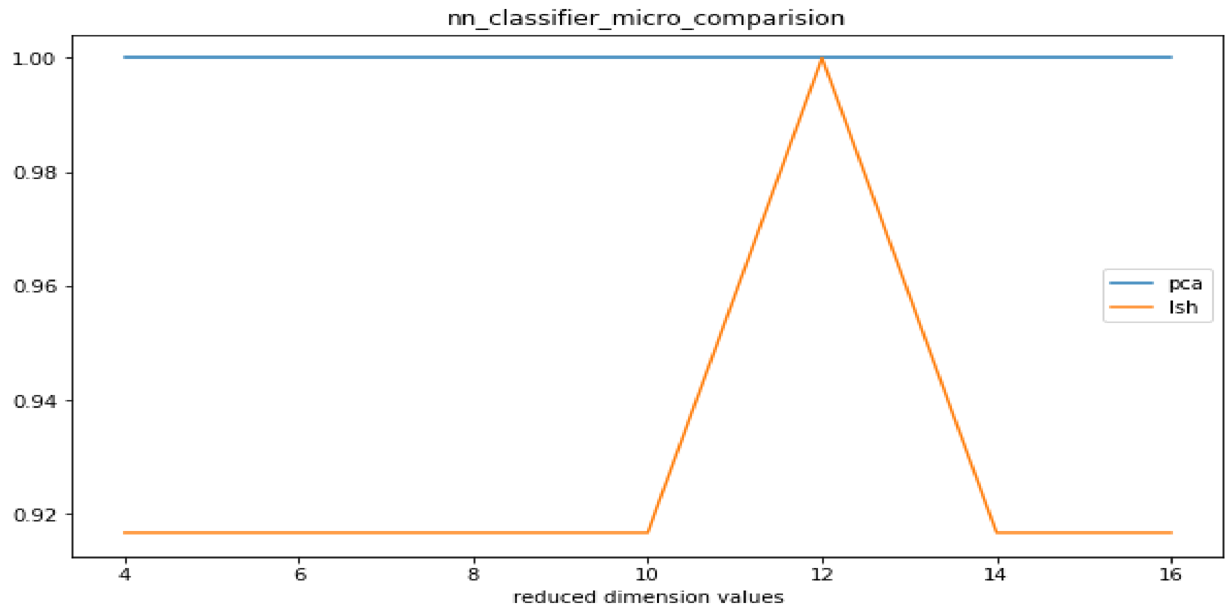
Code File : LSH.py

Implementation Details:

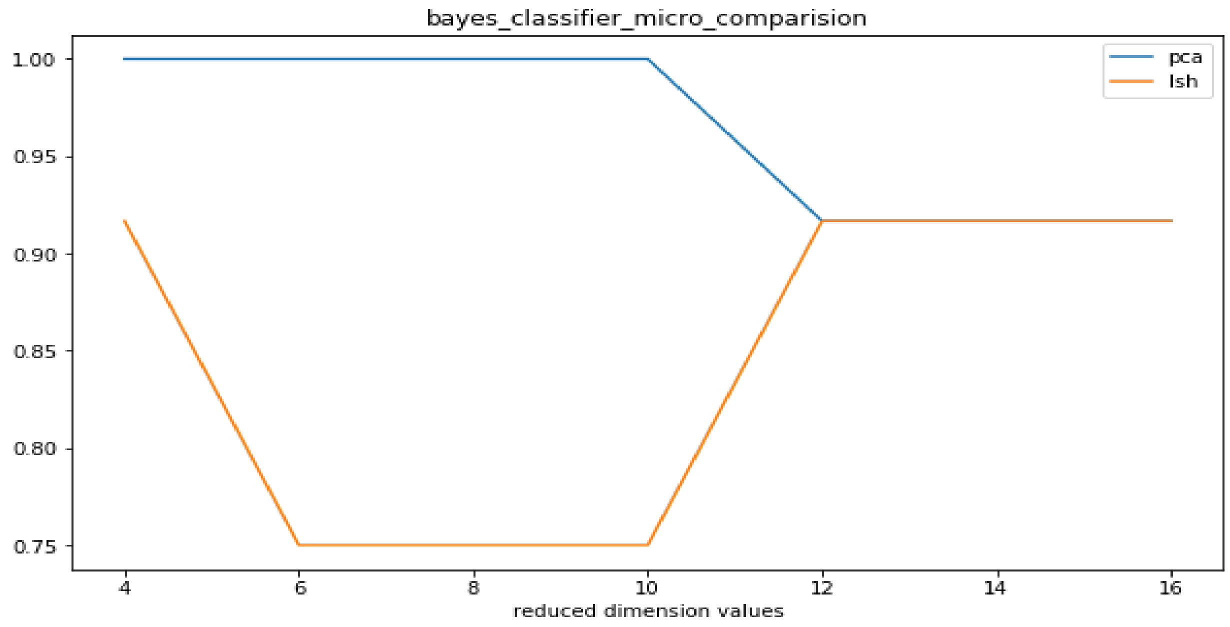
1. First we consider some random number (input by user, say it k) of random hyperplanes.
2. Every data point is projected(dot product) on these hyper planes. The dot product value is binarized, if the value was greater than zero, then I replace it with 1, else by 0. So now each data point is a k dimensional vector.
3. Now this k dimensional vector is divided into bands containing smaller number of partitions. Let it be m bands, each containing n values. So m*n=k.
4. Each band is hashed(classical hashing : binary to decimal conversion). Data points having similar hash values in any band are considered to be a member for nearest neighbor.
5. Here the task is to reduce the dimension. So I have used the m bands hash values as the reduced dimension values. m is the input by user and I have fixed n=8. So now each data point has been converted to m dimensional vector.

TASK 7 LSH and PCA comparision:

Dolphins dataset, NN Comparision

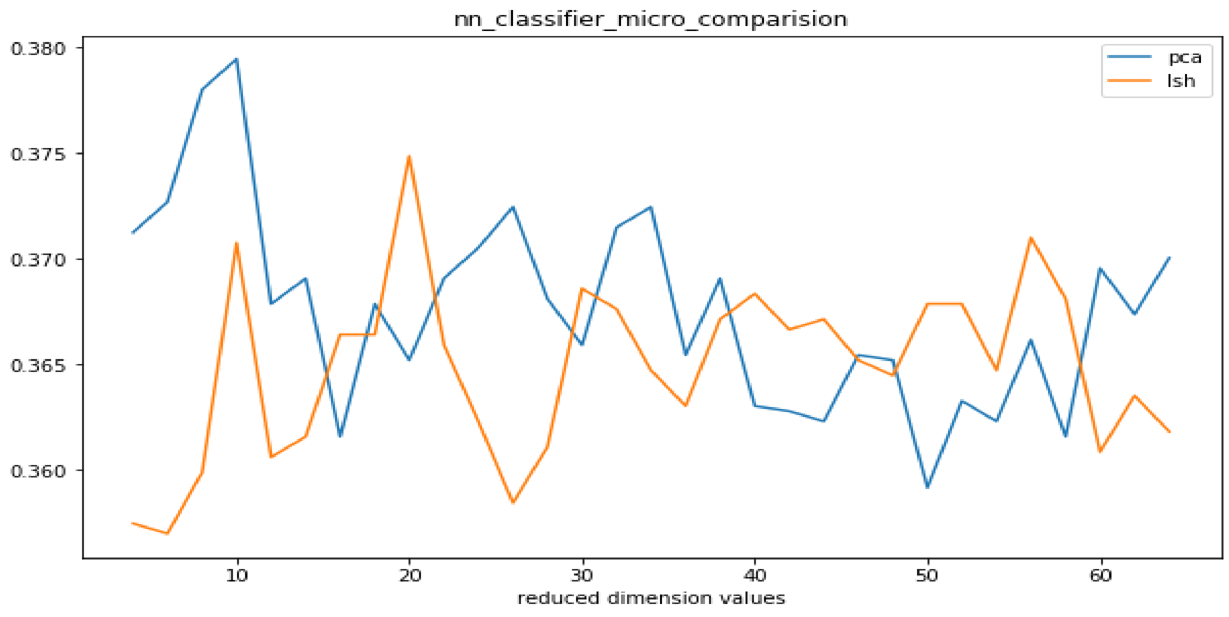


Bayes Comparision

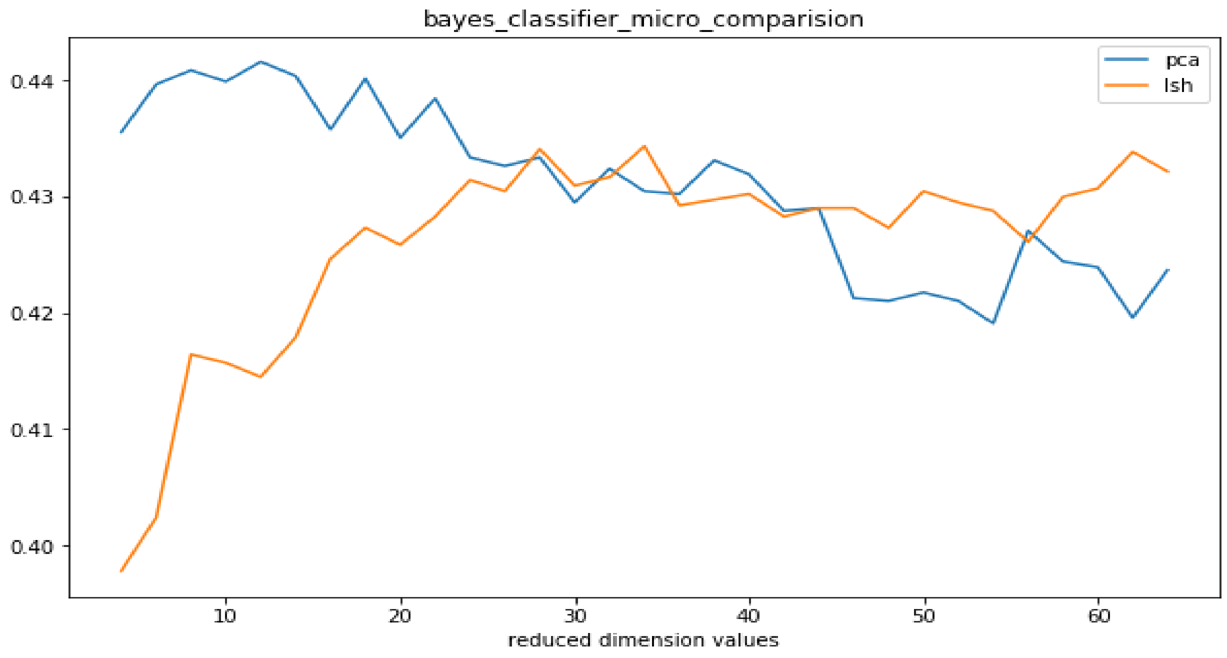


In both the cases LSH is giving poor performance. Point to note is performance of PCA in BAYES classifier has decreased with increase in dimension.

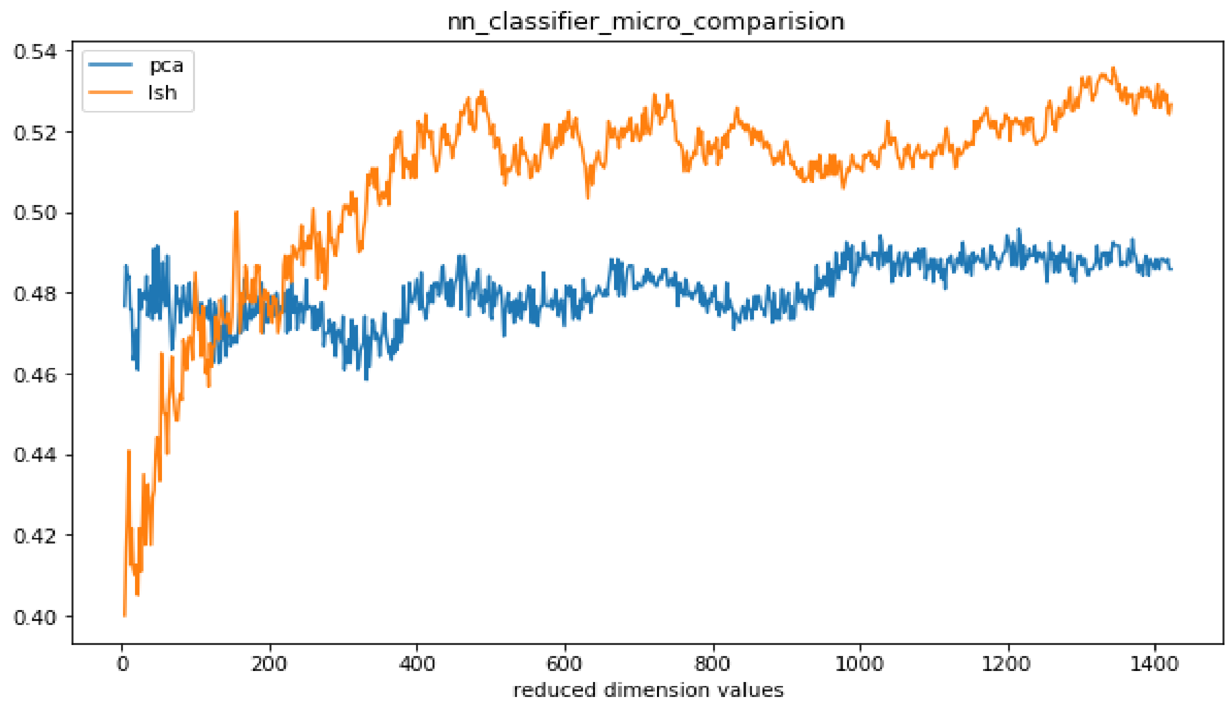
PubMed Dataset, NN comparision: At some places LSH is giving better performance than PCA. Performance of LSH is increasing with increase in dimension, whereas for PCA its decreasing both cases.For NN, there is no much improvement in performance for LSH with increase in dimensionality but for Bayes performance is increasing.



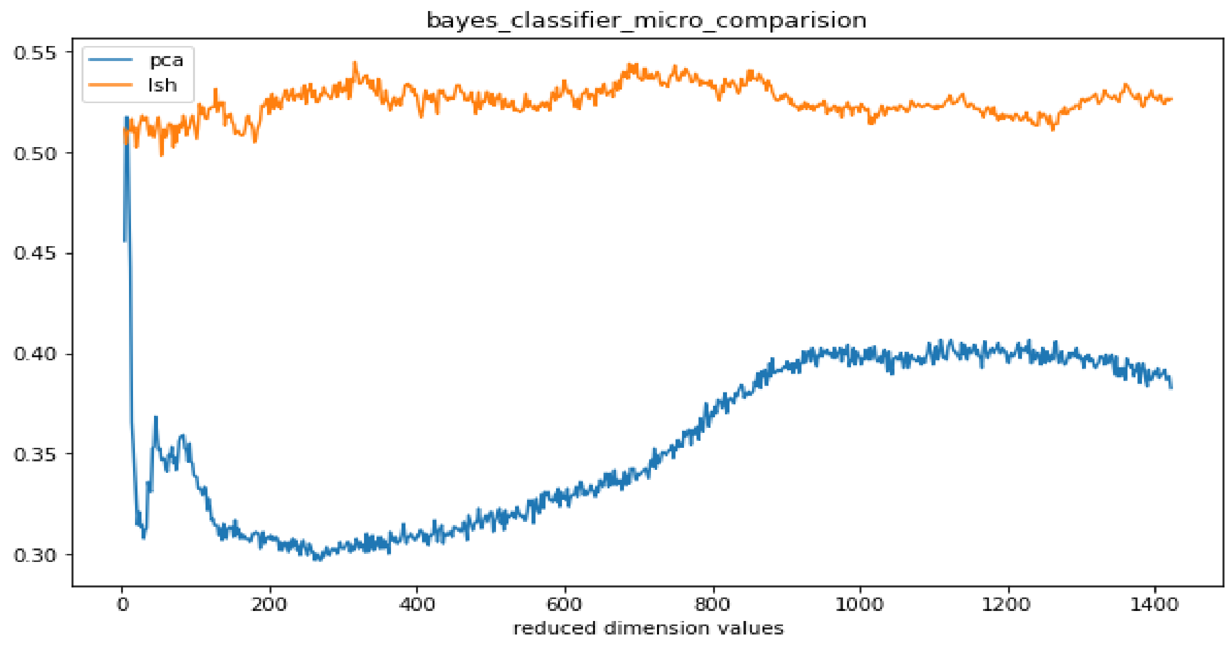
PubMed, Bayes Comparision:



Twitter Dataset: For NN Performance of LSH increases with increase in dimensions.



Bayes Comparision



LSH is giving better performance for Bayes classifier in this dataset as compared to PCA.