ASSIGNMENT – 1

Submitted by: Akshaya A Mukundan
SR No. 15521

Task - I (10 Points)

Implement Random Projections algorithm to convert the high-dimensional data into lower dimension. If the original data dimension is K, your algorithm should be able to lower it down to D(D < K). Experiment with D = 2; 4;….[K/2] for all the datasets given and save them into _les of suitable format for later use. DO NOT use any python library. Implement on your own.

Algorithm:

To reduce the dimension of the given dataset, the dataset is multiplied by a matrix of suitable dimension, the result of which is the reduced dimensional data.

1) Fetch the data in csv file. For twitter dataset vectorize the tweets and then store these data in a list. The name of the list is 'dataset' in the code.
2) Fetch the labels corresponding to data. The labels are stored in 'list1'.
3) The size of dataset is Number of datasets(rows) x Number of features. Hence, multiply the dataset by a random matrix of size Number of features x Reduced dimension required, to get the low dimensional matrices.
4) Store the low dimensional matrices again in csv format for ease in use in further tasks.

Results:

Reduced dimensions of all datasets are obtained.
Dolphin dataset: Feature size = 32
Reduced dimensions are 2, 4, 8, 16 and 32
Pubmed dataset: Feature size = 128
Reduced dimensions are 2, 4, 8, 16 and 32, 64
Twitter dataset: Feature size = 2935
Reduced dimensions are 2,4,8,16,32,64,128,256,512 (Dimensions of 1024 couldn't be obtained due to Memory error (I don't have access to cls server)

Task - II (15 Points)

Design Bayes classifier and Nearest Neighbour classifier. For Bayes classifier, you need to estimate class conditional densities (follow a suitable scheme e.g. maximum likelihood) using training data. You'll be using both of these for the classification problem on the datasets provided herewith. You are not allowed to use any package. Design from scratch.

Algorithm(kNN):

1) Fetch the data in csv file. For twitter dataset vectorize the tweets and then store these data in a list and also save it as a csv file for further tasks. The name of the list is 'dataset' in the code.
2) Fetch the labels corresponding to data. The labels are stored in 'list1'.
3) Using k fold cross validation, separate each $1/10^{th}$ of dataset into test set and the remaining into trainingset in 10 iterations.
4) In each iteration, find the Euclidean distance between each vector in the test set and all the vectors in the training set. Store K =5 nearest neighbors of each test vector.
5) Find the class of these 5 nearest neighbors. Maximum of these gives the class likely to be the class of the test vector. Similarly, do for all the test vector and for k iterations of the k fold cross validation.
6) To measure accuracy, find the number of correct judgements divided by total number of judgements
7) Print classification report to get the F1 score

Results:

Implemented kNN algorithm from scratch is found to be working for all datasets with sufficient accuracy. Output prints accuracy and F1 scores.

Algorithm(Naïve Bayes Classifier):

1) Fetch the data in csv file. For twitter dataset vectorize the tweets and then store these data in a list and also save it as a csv file for further tasks. The name of the list is 'dataset' in the code.
2) Fetch the labels corresponding to data. The labels are stored in 'list1'.
3) Using k fold cross validation, separate each $1/10^{th}$ of dataset into test set and the remaining into trainingset in 10 iterations.
4) For all the vectors in the training set falling in same class, find the mean across each column. And then find the variance.
5) For each test vector, using Gaussian probability distribution, find the probability(Bayes algorithm-maximum likelihood) that the test vector falls

in a particular class given the features of the test vector. Repeat this for each class.

6) The class which gives the maximum of these probability is likely to be the class of the test vector. Similarly, do for all the test vector and for k iterations of the k fold cross validation.
7) To measure accuracy, find the number of correct judgements divided by total number of judgements
8) Print classification report to get the F1 score

Results:

Implemented Naïve Bayes algorithm from scratch is found to be working for all datasets with sufficient accuracy. Output prints accuracy and F1 scores.

Task - III (10 Points)

Divide the data (both the original/high and low-dimensional) into train and test set using cross-validation technique. Measure accuracy and F1-score (Macro and Micro) for all the data sets. Plot the results for different values of D and K. What is your take-away? Which side are you in - Amar's or Akbar's?

Algorithm:

1) Use the k-NN and Naïve Bayes algorithm implemented in the Task 2. Fetch the original datasets and find the accuracy and F1 score.
2) Repeat the same for all the low dimensional data obtained using random projections(task 1)

Results:

Performed k fold cross validation (k = 10) for all the datasets to split into train and test sets. Performed kNN and Naïve Bayes algorithm. Measure accuracy and F1 score for all the datasets and their low dimensional data.

Dolphins**:**

	K = 32	D = 16	8	4	2
Accuracy	0.9286	0.9286	0.8857	0.9428	0.8143
F1 Score (micro)	0.93	0.93	0.89	0.94	0.81
F1 Score (macro)	0.88	0.89	0.82	0.90	0.77

Pubmed:

	K = 128	D = 64	32	16	8	4	2
Accuracy	0.38	0.3807	0.3824	0.3711	0.3735	0.3714	0.38
F1 Score (micro)	0.37	0.38	0.38	0.37	0.37	0.38	0.38
F1 Score (macro)	0.33	0.34	0.35	0.34	0.34	0.34	0.35

Twitter:

	K = 2935	D = 128	64	32	16
Accuracy	0.509	0.4493	0.4545	0.4412	0.4473
F1 Score (micro)	0.51	0.45	0.45	0.44	0.45
F1 Score (macro)	0.44	0.36	0.35	0.33	0.34

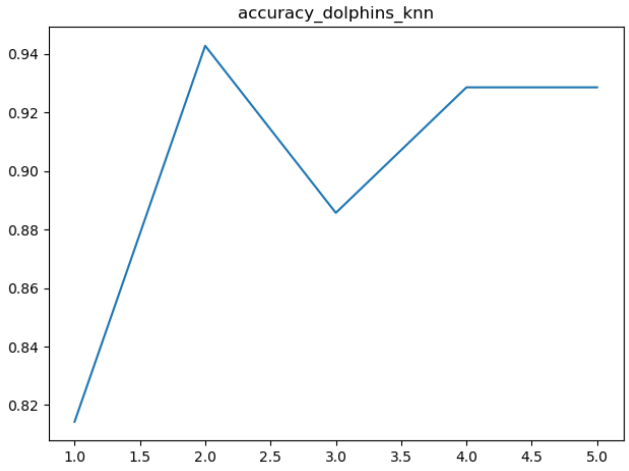
	D = 8	D = 4	2
Accuracy	0.4315	0.4548	0.4353
F1 Score (micro)	0.43	0.45	0.44
F1 Score (macro)	0.34	0.35	0.33

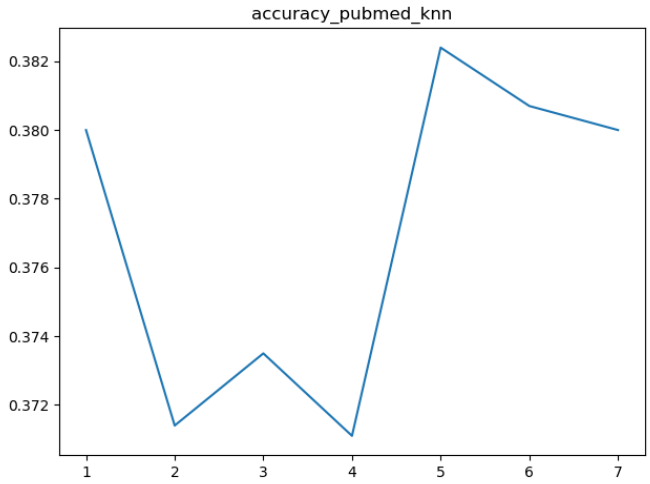
Inference:

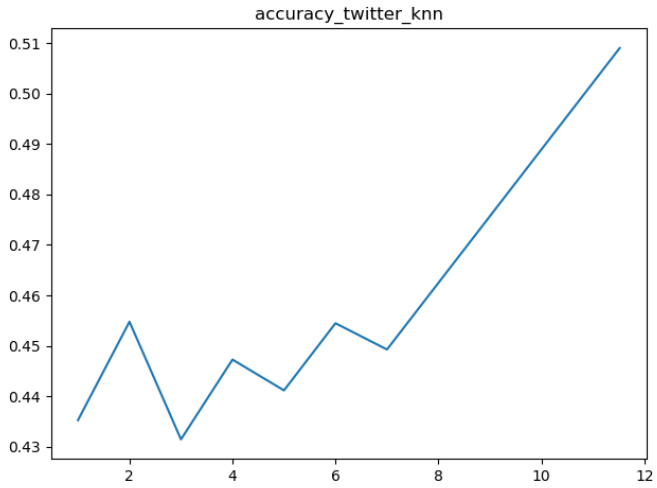
As the dimension increases, accuracy is found to increase mildly. Hence, Amar is more likely to be true.

But in general there is no such pattern that as the dimension increases accuracy and F1 score would increase or decrease. Hence, both Amar and Akbar are wrong.

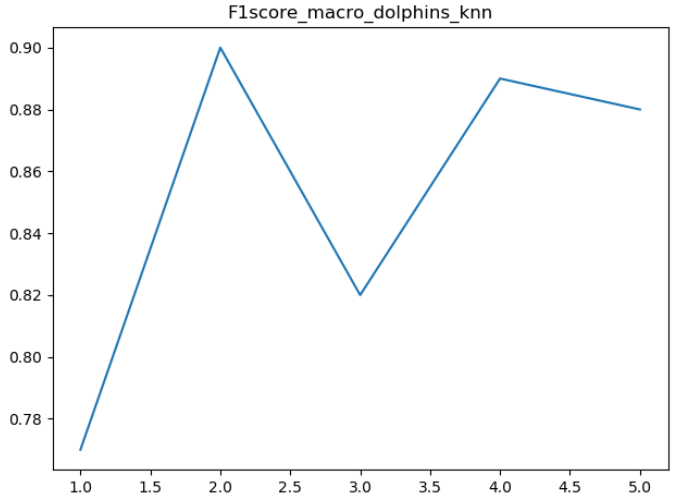
Accuracy plots: (Plots have x axis in $\log_2$ format. i.e 2 is represented as 1, 32 => 5)

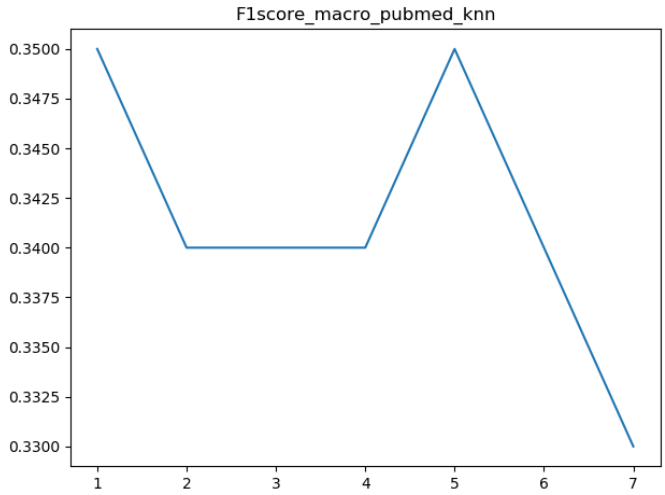


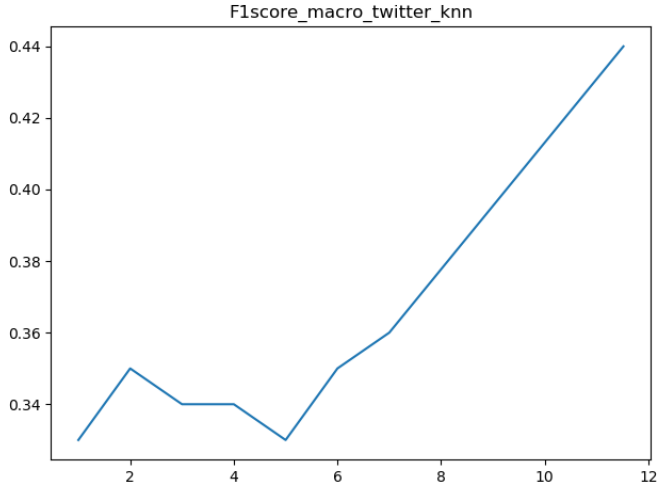




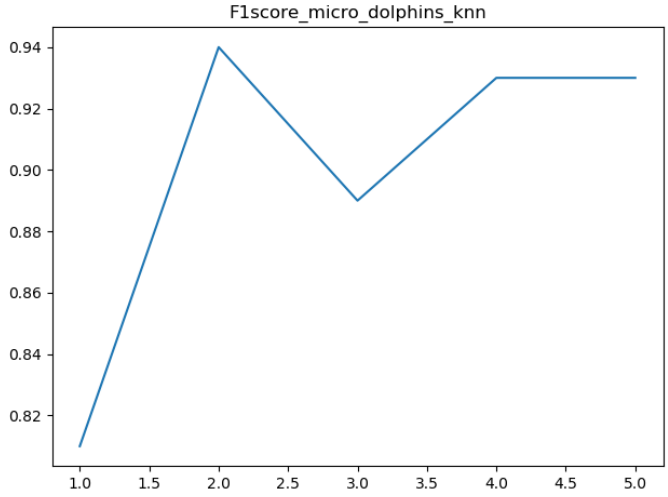
F1 score (macro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

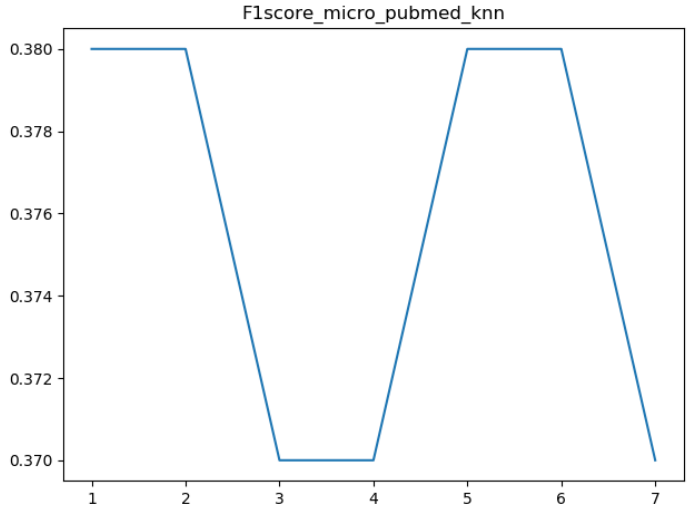


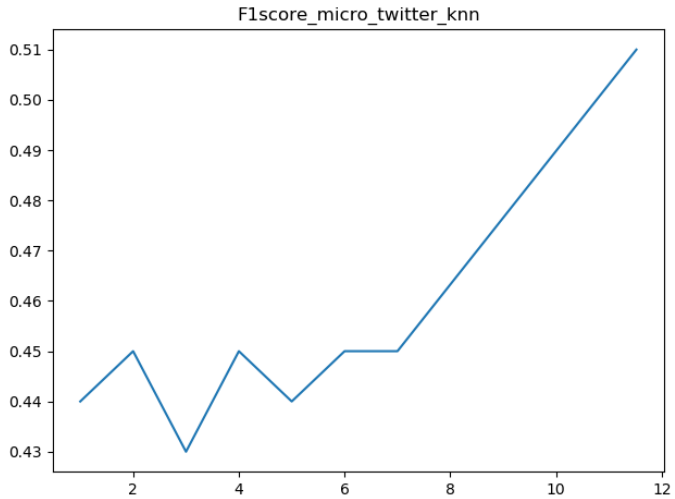




F1 score (micro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)







Naïve Bayes Results:

Dolphins**:**

	K = 32	D = 16	8	4	2
Accuracy	0.91	0.8066	0.8567	0.9152	0.7952
F1 Score (micro)	0.92	0.82	0.83	0.92	0.79
F1 Score (macro)	0.85	0.68	0.71	0.78	0.61

Pubmed:

	K = 128	D = 64	32	16	8	4	2
Accuracy	0.395	0.42	0.4172	0.41	0.42	0.42	0.414
F1 Score (micro)	0.39	0.41	0.42	0.41	0.42	0.42	0.41
F1 Score (macro)	0.37	0.31	0.31	0.31	0.31	0.31	0.31

Twitter:

	K = 2935	D = 128	64	32	16
Accuracy	0.2562	0.3369	0.4004	0.4251	0.4579
F1 Score (micro)	0.26	0.34	0.4	0.43	0.47
F1 Score (macro)	0.14	0.32	0.35	0.31	0.29

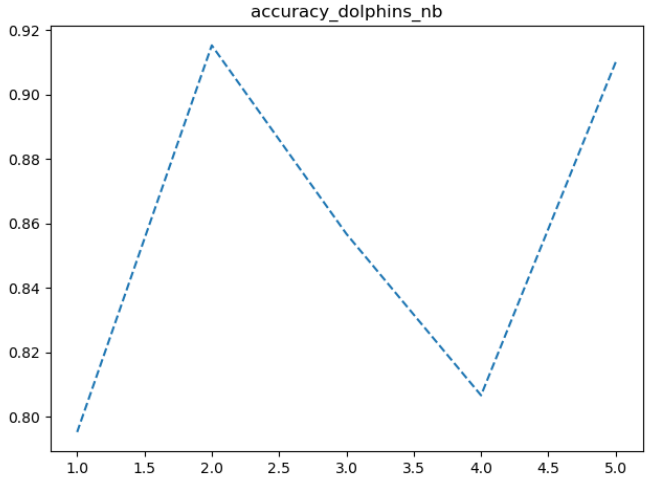
	D = 8	D = 4	2
Accuracy	0.4791	0.4904	0.4978
F1 Score (micro)	0.50	0.50	0.51
F1 Score (macro)	0.27	0.25	0.23

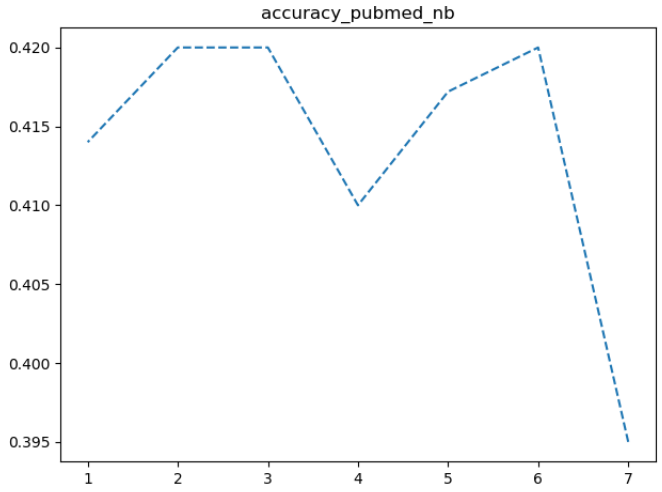
Inference:

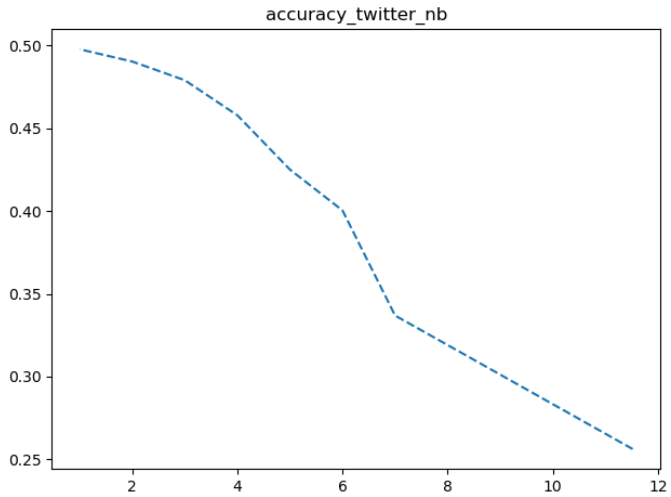
As the dimension increases, accuracy is found to increase mildly. Hence, Amar is more likely to be true.

But in general there is no such pattern that as the dimension increases accuracy and F1 score would increase or decrease.

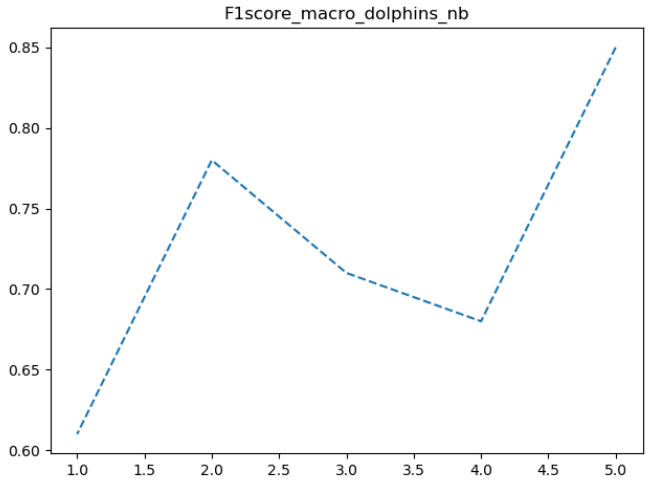
Accuracy: (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

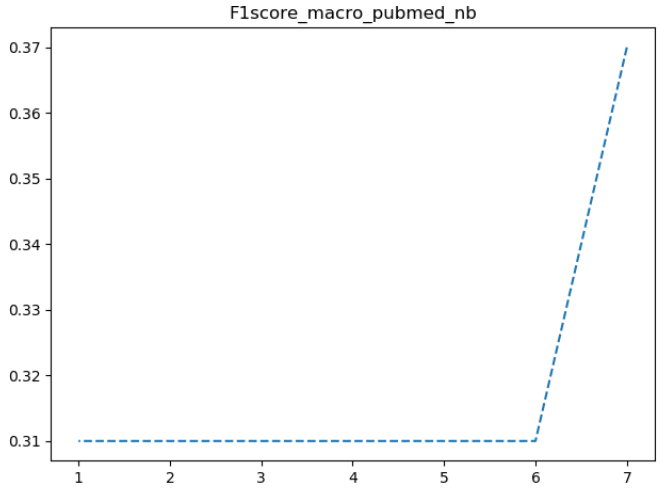


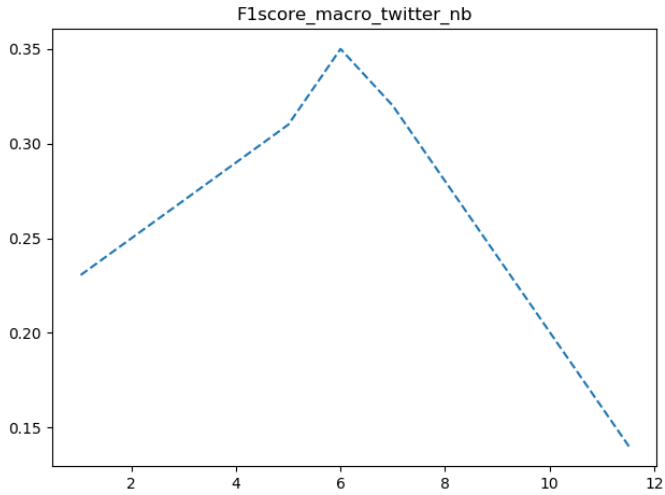




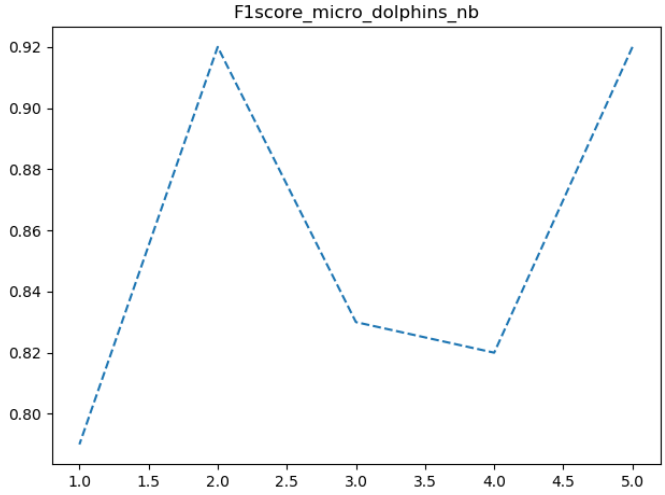
F1 Score (macro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

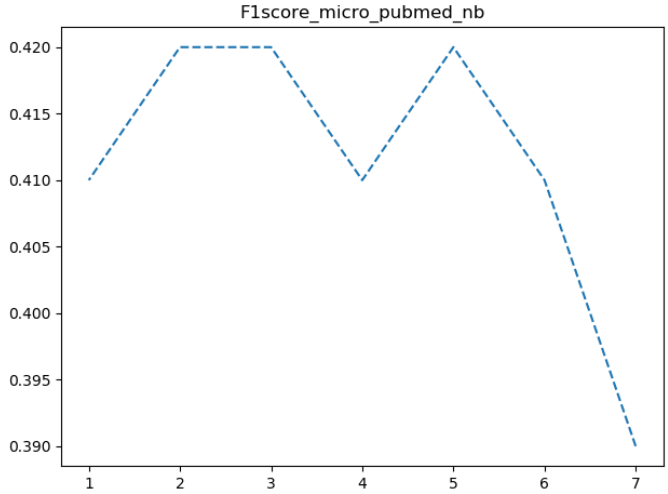


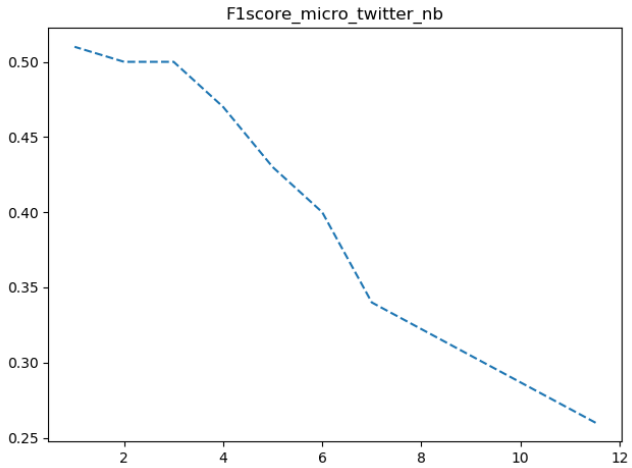




F1 Score (micro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)







Task - IV (5 Points)

Do the same as in task-III, but this time use scikit-learn library for both Bayes and NN classifier.

Algorithm:

1) Use the k-NN and Naïve Bayes algorithm in the scikit-learn library. Fetch the original datasets and find the accuracy and F1 score.
2) Repeat the same for all the low dimensional data obtained using random projections(task 1)

Results:

Performed k fold cross validation(k = 10) for all the datasets to split into train and test sets. Performed kNN and Naïve Bayes algorithm using sklearn library. Measured accuracy and F1 score for all the datasets and their low dimensional data.

kNN Results:

Dolphins**:**

	K = 32	D = 16	8	4	2
Accuracy	0.9285	0.929	0.9143	0.9428	0.8143
F1 Score (micro)	0.93	0.93	0.91	0.94	0.81
F1 Score (macro)	0.88	0.89	0.82	0.90	0.77

Pubmed:

	K = 128	D = 64	32	16	8	4	2
Accuracy	0.3725	0.3714	0.3657	0.3643	0.36	0.3627	0.3559
F1 Score (micro)	0.37	0.37	0.37	0.36	0.36	0.36	0.36
F1 Score (macro)	0.35	0.35	0.34	0.34	0.34	0.34	0.34

Twitter:

	K = 2935	D = 128	64	32	16
Accuracy	0.4988	0.4451	0.4293	0.4149	0.4174
F1 Score (micro)	0.50	0.45	0.43	0.41	0.42
F1 Score (macro)	0.42	0.36	0.36	0.34	0.34

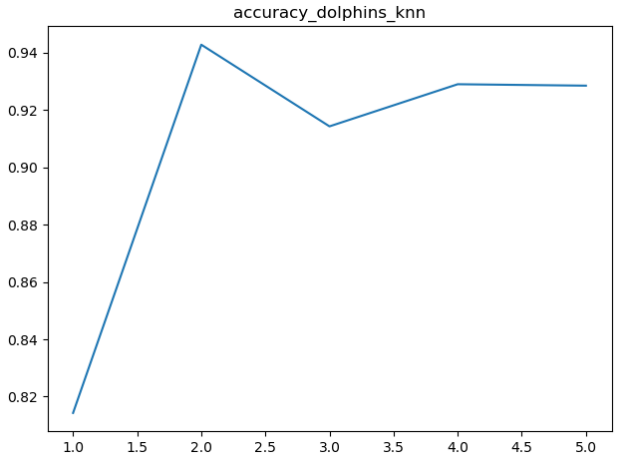
	D = 8	D = 4	2
Accuracy	0.4057	0.4254	0.4068
F1 Score (micro)	0.41	0.43	0.41
F1 Score (macro)	0.33	0.35	0.33

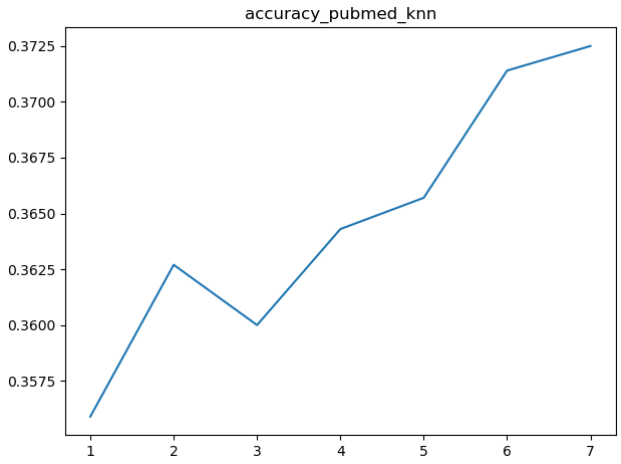
Inference:

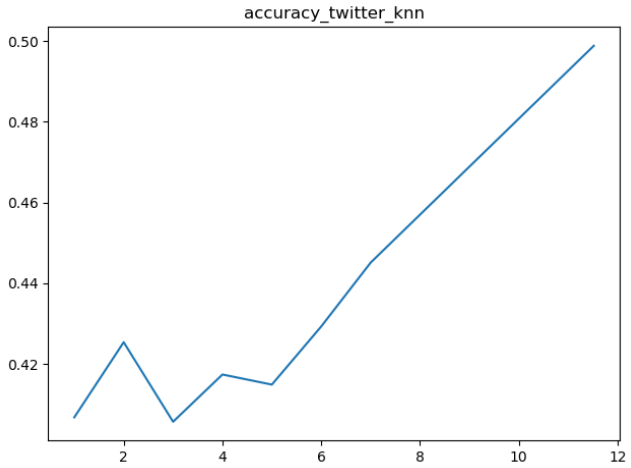
As the dimension increases, accuracy is found to increase mildly. Hence, Amar is more likely to be true.

But in general there is no such pattern that as the dimension increases accuracy and F1 score would increase or decrease.

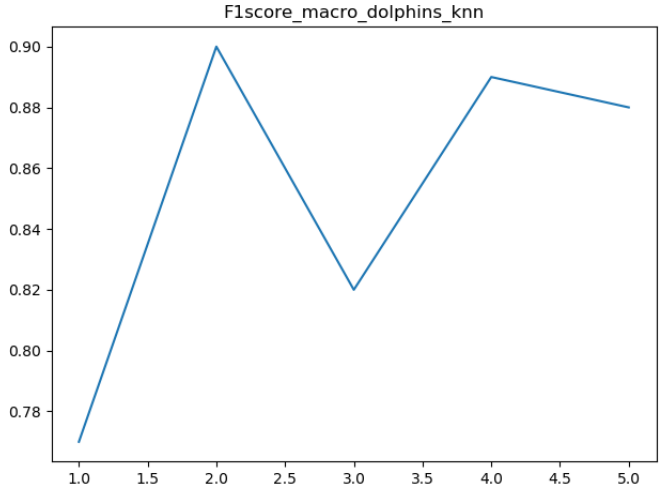
Accuracy: (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

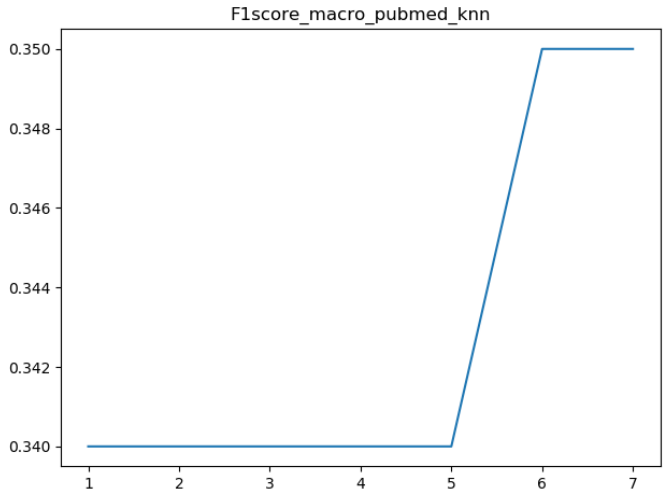


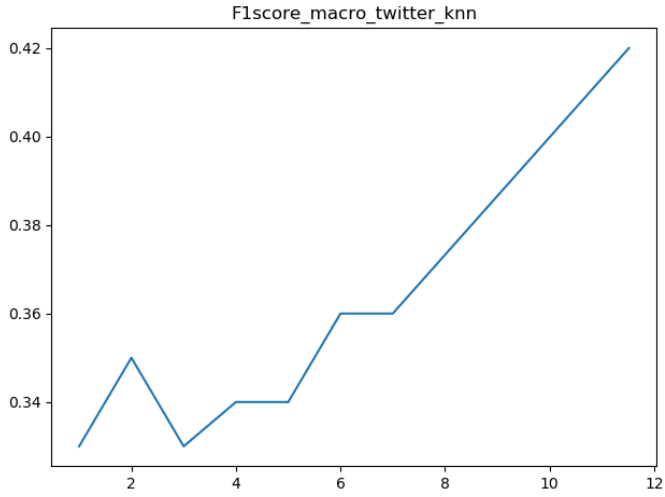




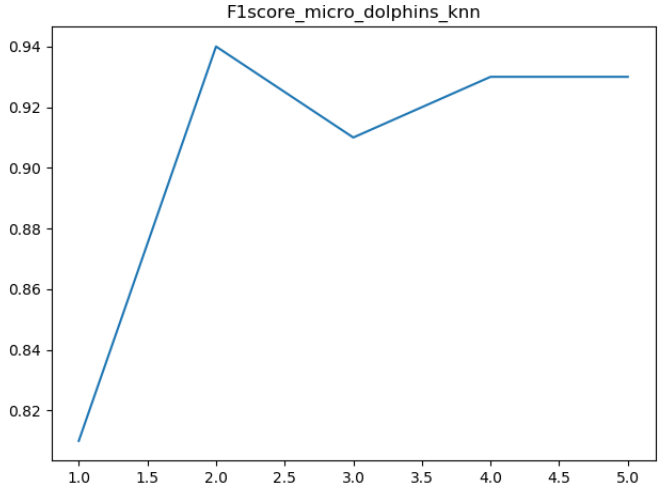
F1Score (Macro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

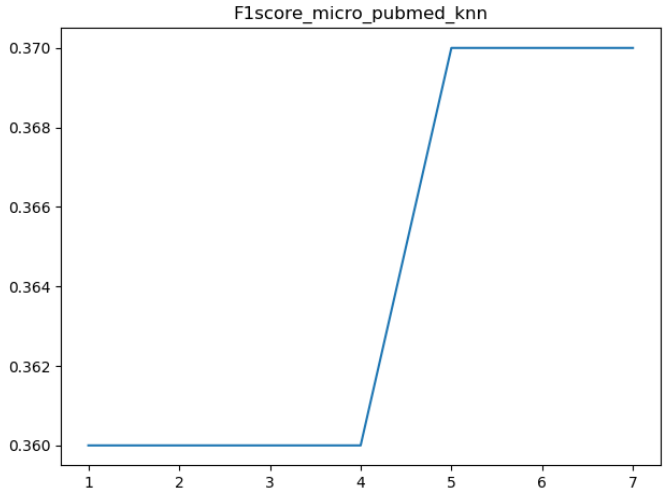


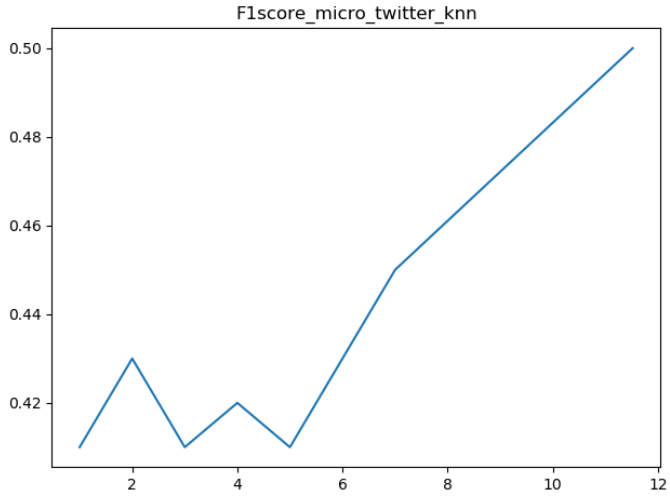




F1Score (Micro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)







Naïve Bayes Results:

Dolphins**:**

	K = 32	D = 16	8	4	2
Accuracy	0.90	0.7571	0.8286	0.9014	0.8143
F1 Score (micro)	0.90	0.76	0.83	0.90	0.82
F1 Score (macro)	0.86	0.62	0.71	0.78	0.68

Pubmed:

	K = 128	D = 64	32	16	8	4	2
Accuracy	0.4238	0.4170	0.4180	0.4134	0.4246	0.4155	0.4132
F1 Score (micro)	0.42	0.42	0.42	0.41	0.42	0.42	0.41
F1 Score (macro)	0.40	0.31	0.31	0.31	0.31	0.31	0.31

Twitter:

	K = 2935	D = 128	64	32	16
Accuracy	0.3768	0.3376	0.4107	0.4435	0.4883
F1 Score (micro)	0.38	0.34	0.41	0.44	0.49
F1 Score (macro)	0.34	0.32	0.35	0.32	0.29

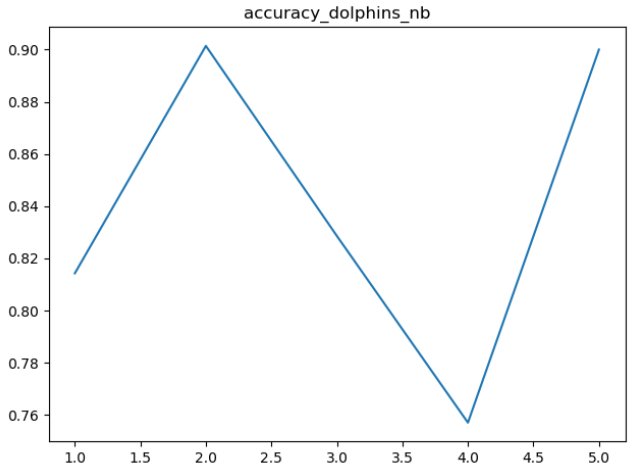
	D = 8	D = 4	2
Accuracy	0.5056	0.5098	0.5169
F1 Score (micro)	0.51	0.51	0.52
F1 Score (macro)	0.29	0.25	0.23

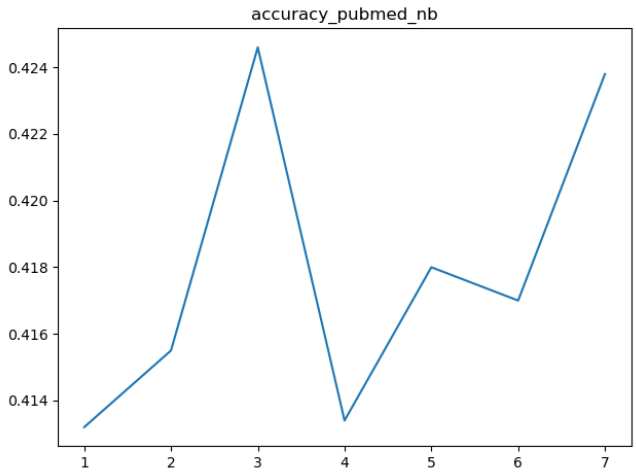
Inference:

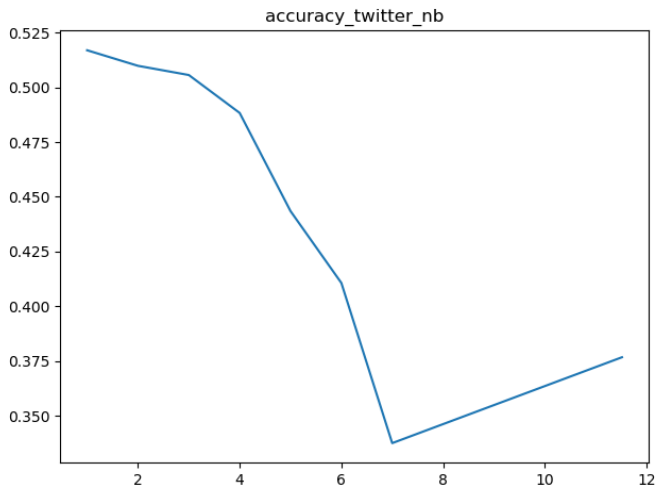
As the dimension increases, accuracy is found to increase mildly. Hence, Amar is more likely to be true.

But in general there is no such pattern that as the dimension increases accuracy and F1 score would increase or decrease.

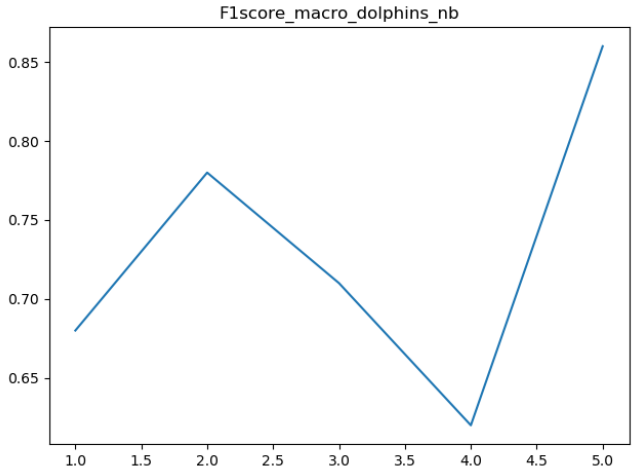
Accuracy: (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

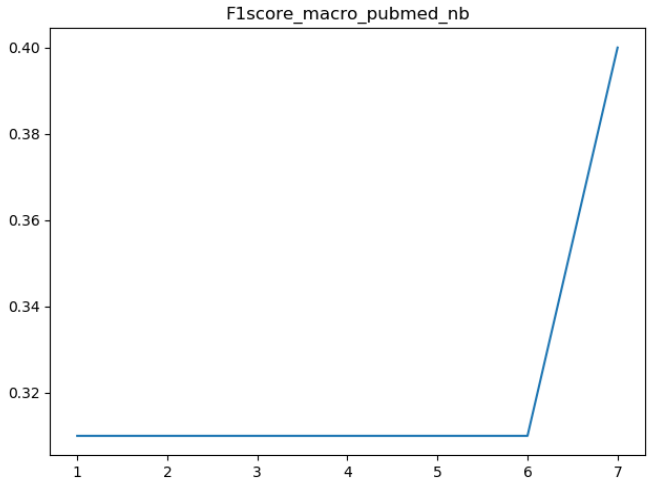


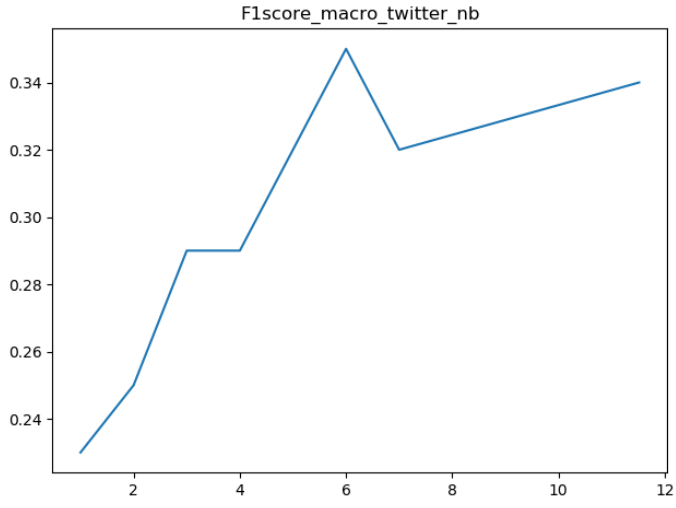




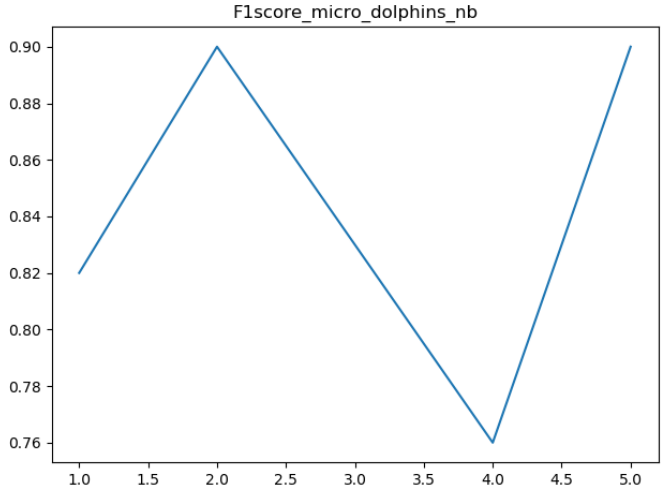
F1 Score (macro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

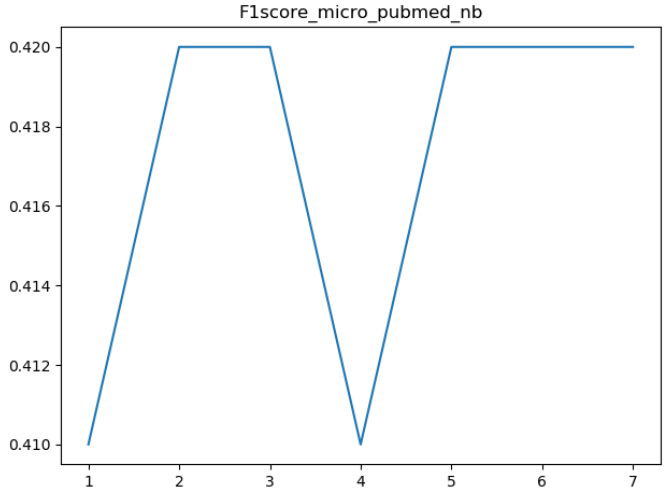


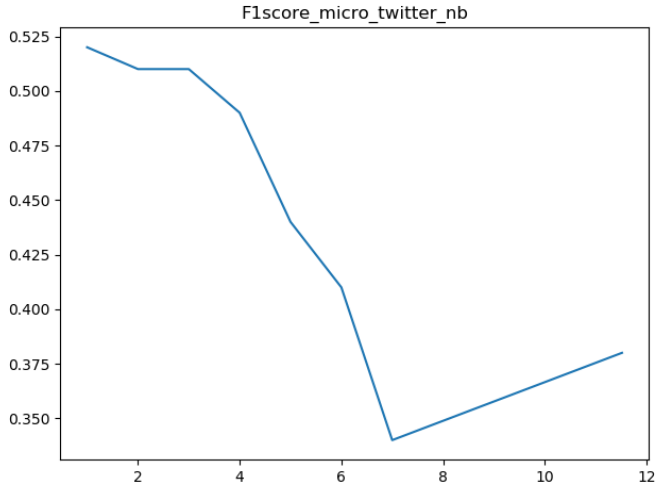




F1 Score (micro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)







Task - V (5 Points)

Compare the results obtained in Task-III and Task-IV. Plot the comparisons accordingly. Is your classifier able to outperform scikit-learn's? If not, what might be the possible reasons you can think of?

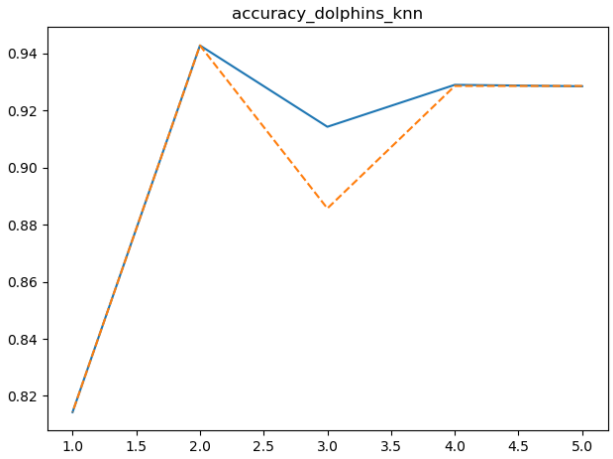
Comparison plots:

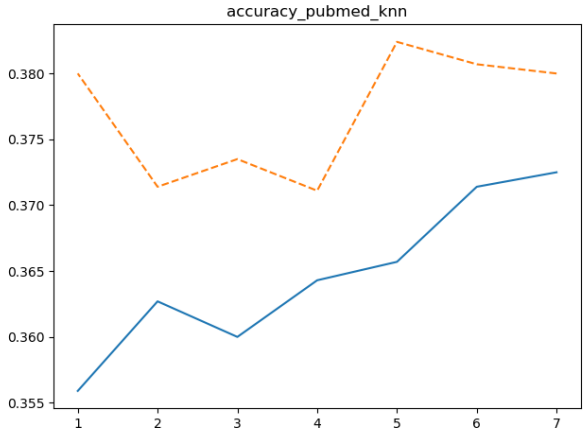
Dotted orange plot: kNN implemented from scratch

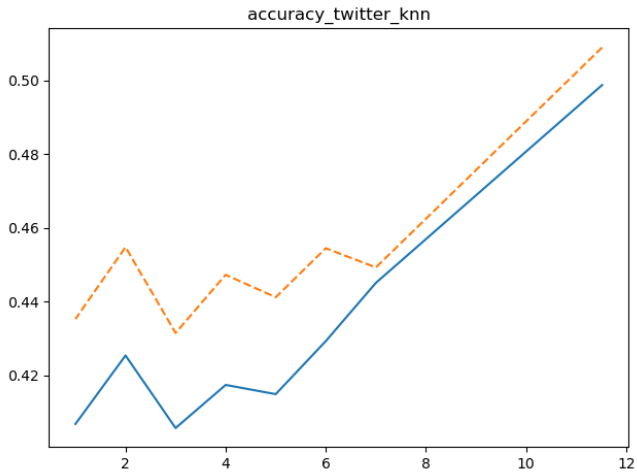
Blue line: kNN from scikit-learn library

Result comparison of kNN:

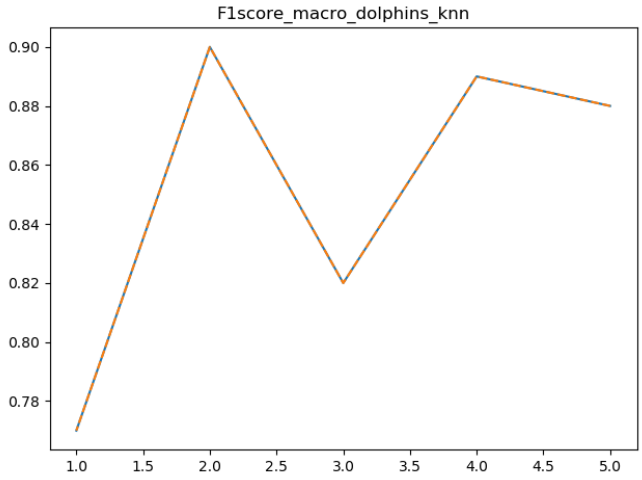
Accuracy: (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

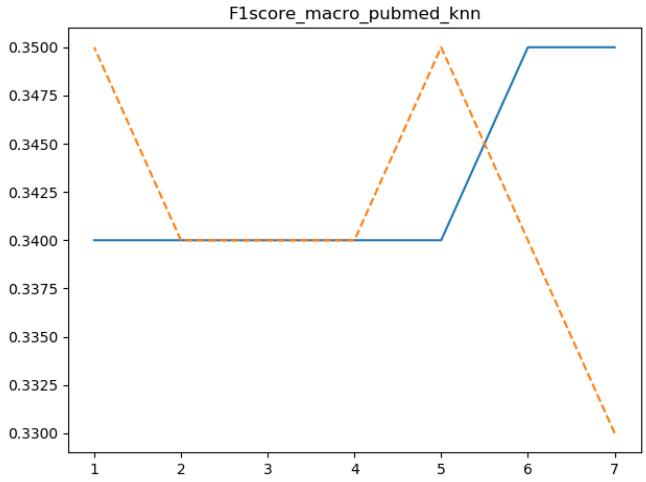


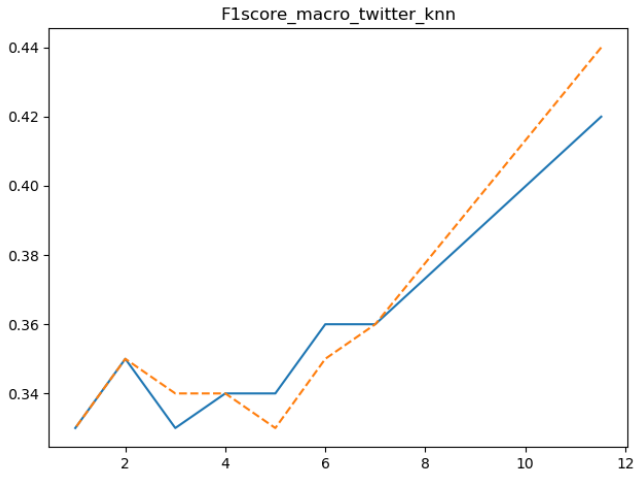




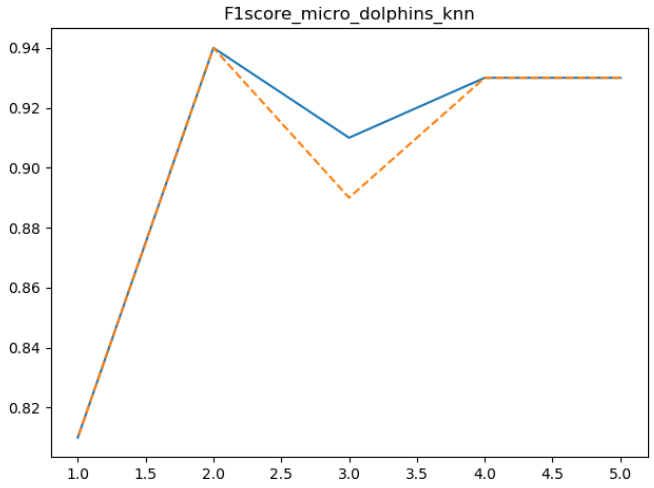
F1 Score (macro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

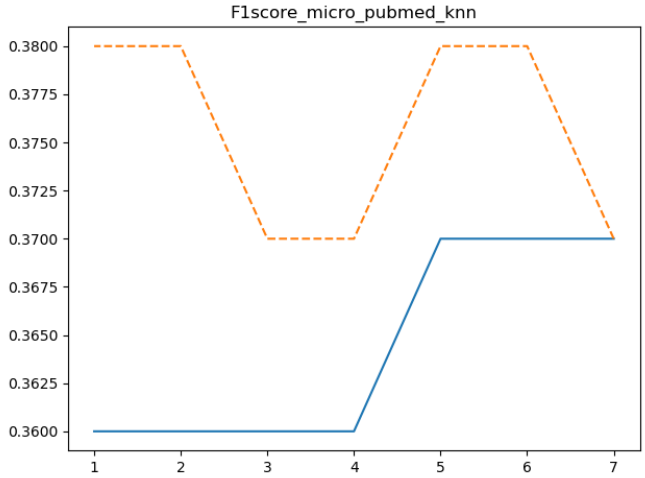


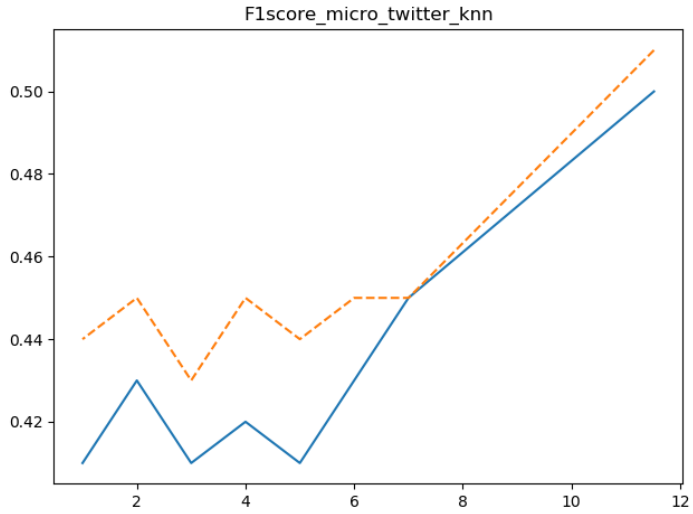




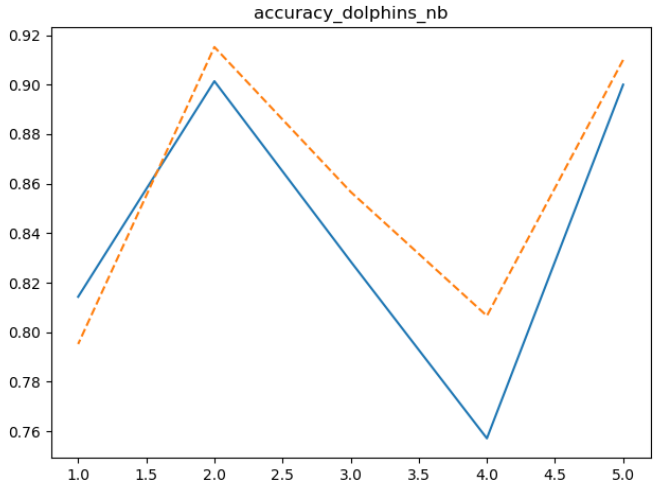
F1 Score (micro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

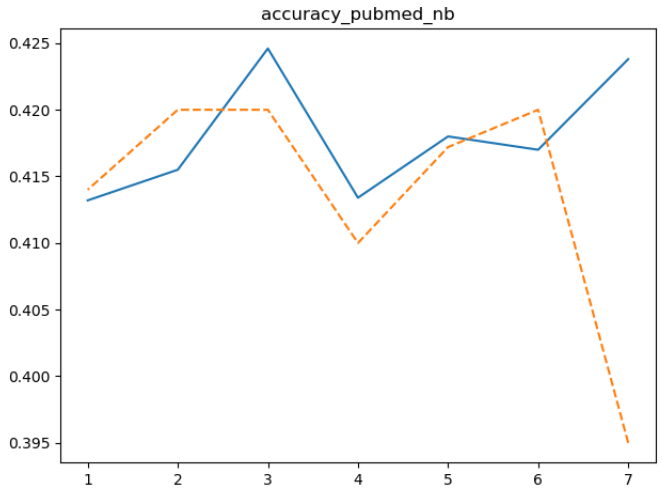


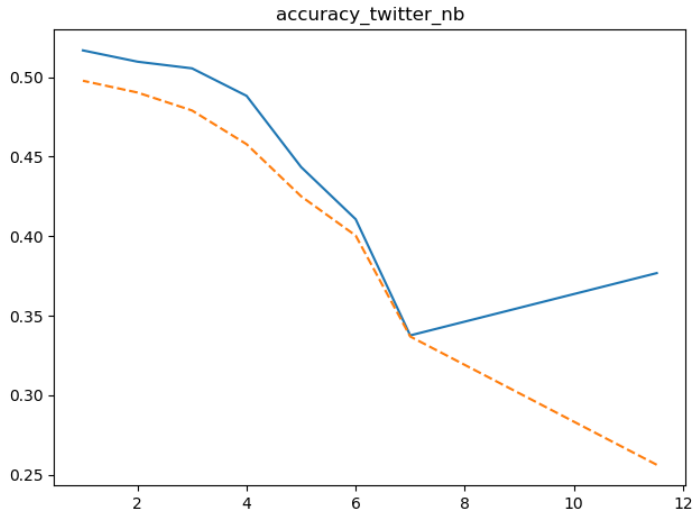




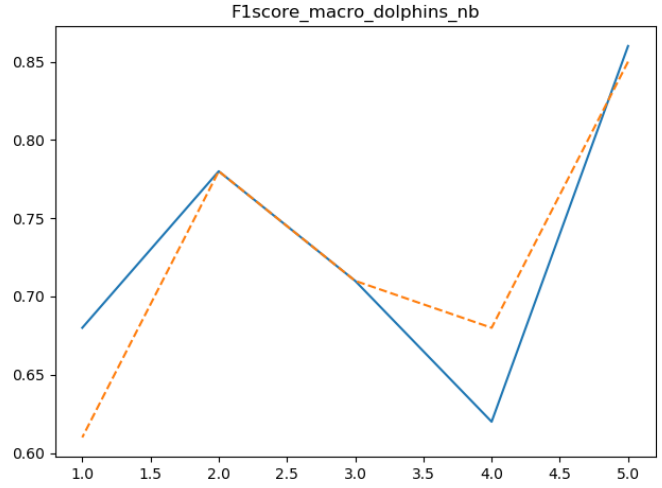
Result comparison of Naïve Bayes:
Accuracy: (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

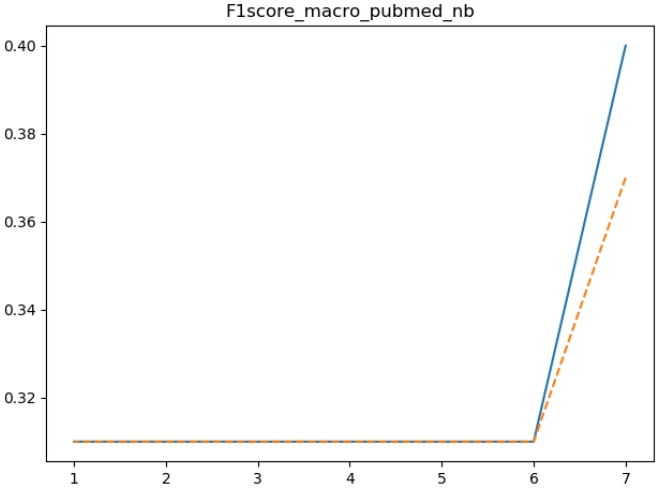


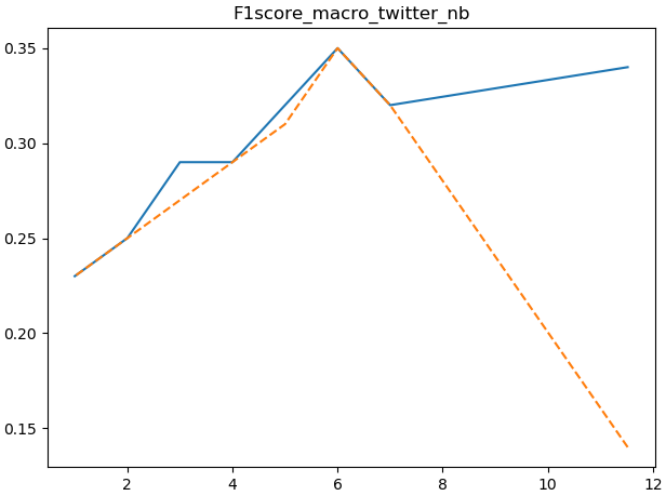




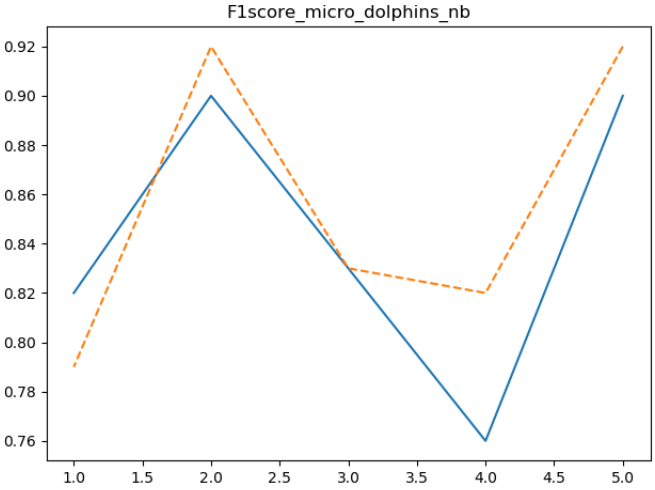
F1 Score (macro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

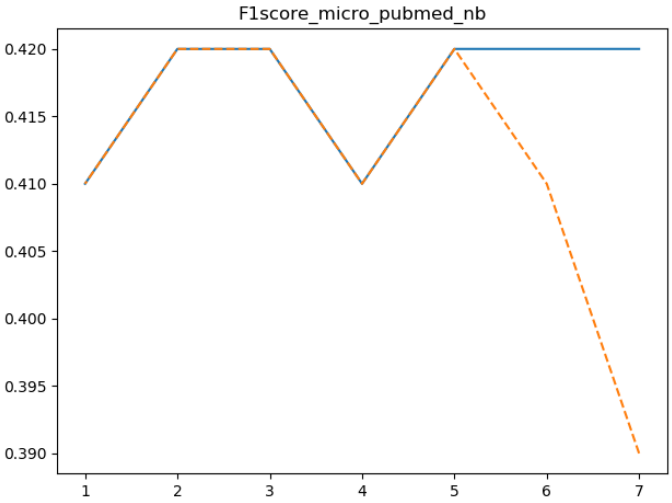


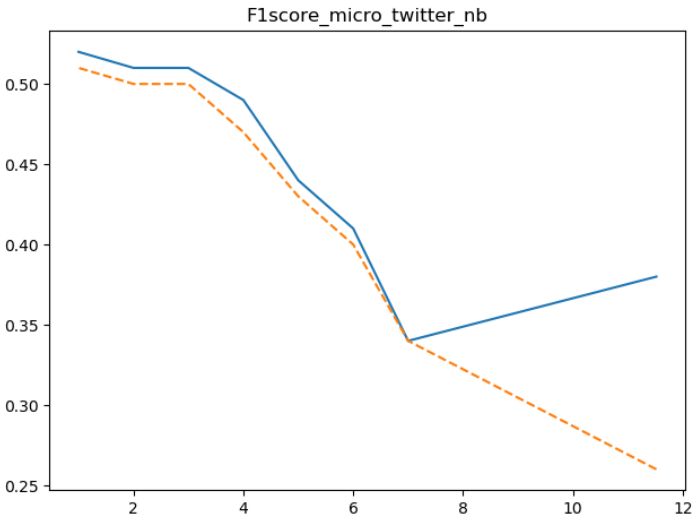




F1 Score (micro): (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)







Comparison inference:

kNN:

Accuracy: Implemented code accuracy drops mildly in intermediate dimensions for dolphin. Whereas for pubmed and twitter, implemented code achieves higher accuracy than the scikit learn code.

F1 Score (Macro): Implemented code is almost parallel to that of the one implemented using sklearn library. Although mild changes are observed in case of pubmed and twitter dataset.

F! Score (Micro): Implemented code is almost parallel to that of the one implemented using sklearn library for dolphin dataset. But, for pubmed and twitter dataset, the one implemented from scratch is found to have higher accuracy.

Naïve Bayes:

Accuracy: Implemented code from scratch has higher accuracy for dolphin dataset, whereas accuracy is almost same for pubmed and twitter dataset.

F1 Score (Macro): It's found to be almost same for all the 3 datasets.

F1 Score (Micro): It's found to be almost same for all the 3 datasets.

The reason for the mild change in accuracy might be due to difference in the algorithm. The inbuilt library algorithm is found to be more accurate for higher dimensions, which is what is required for practical applications as the dataset and feature size is generally high in practical application scenarios.

Task - VI (10 Points + 15 Bonus)
Implement Locality Sensitive Hashing(LSH) to reduce the dimensionality of the data. Preferably, do it on your own. Bonus credits for those who will be doing it independently, without using any library.

Task- VII (10 Points)
Perform the classification task(similar to what you did in Task-III) using the LSH algorithm you developed in the previous question. Compare your results with PCA. You are allowed to use the standard python library for PCA.

Task – VI and VII

Algorithm(LSH and Classification):

1) Fetch the data and assign each value into a list named 'dataset' in the code.
2) Fetch the label set corresponding to the dataset and assign it to a list named 'list1' in the code.
3) Perform k fold cross validation (k = 10) on the dataset to split it into training and test sets, respectively.
4) Fetch each of the values in the training set and convert it into hash tables by multiplying it by a random matrix (which has number of columns as the value to which we would like to reduce the feature size) and assign 1 if the value is greater than zero, else assign zero.
5) Similarly, create hash table of the test vector, the class of which has to be found.
6) Find the hamming distance between the test set and all the hash table of training set row by row. Select the rows of training set that has least hamming distance. Check their classes
7) The maximum class obtained is likely to be the class of the test vector.
8) Repeat the same for all test vectors.

Result:

Implemented LSH from scratch. Separated test and training set using k fold cross validation. Used LSH for creating hash tables.

In LSH, if a point is above a hyper-plane, it's considered as 1, otherwise zero or vice versa. The same concept is used here. Each row of a random matrix defines a hyper-plane. The algorithm assigns a 1 for a positive dot product of vector under

consideration and random matrix, otherwise zero. Hash tables are created in this manner. To reduce the dimensionality, the number of columns in random matrix is the dimension to which we would like to reduce our dataset.

Hence, a reduced dimension matrix of data is obtained using LSH algorithm.

For classification, kNN method is used. Accuracy and F1 Scores are measured for various dimensions of 3 datasets.

Accuracy:

F1 Score (Macro):

F1 Score (Micro):

Algorithm: PCA

1) Datasets are fetched into a list
2) Labels of datas provided are fetched to another list.
3) Using k fold cross validation (k = 10), data is divided into training and test sets.
4) PCA in scikit-learn library is used to perform dimension reduction.
5) Classification is performed on the reduced dimension of data.
6) Accuracy and F1 Score are measured.

Results:

PCA is used for dimensionality reduction and further classification task is performed. Accuracy and F1 Scores are measured for the high and low dimensional data of the 3 datasets.

Accuracy, F1 Score (LSH):

Dolphins:

	D = 16	8	4	2
Accuracy	0.8714	0.7286	0.7143	0.47
F1 Score (macro)	0.8	0.63	0.5	0.31
F1 Score (micro)	0.87	0.73	0.71	0.47

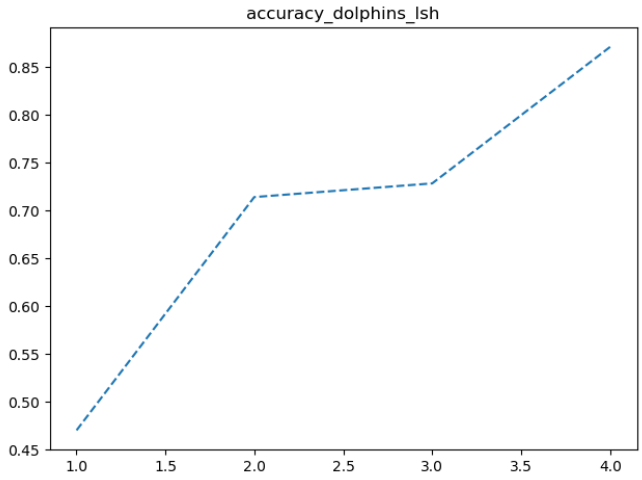
Pubmed:

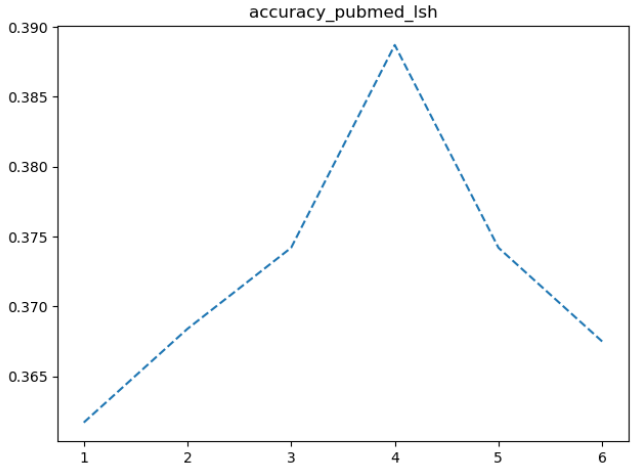
	D = 64	32	16	8	4	2
Accuracy	0.3675	0.3742	0.3887	0.3742	0.3684	0.3617
F1 Score (macro)	0.37	0.37	0.39	0.38	0.37	0.35
F1 Score (micro)	0.34	0.34	0.35	0.33	0.33	0.36

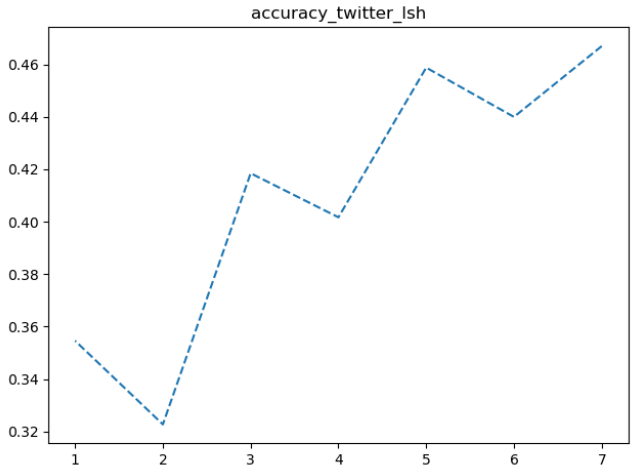
Twitter:

	D = 128	64	32	16	8	4	2
Accuracy	0.467	0.440	0.4588	0.4017	0.4185	0.3227	0.3546
F1 Score (macro)	0.33	0.3	0.32	0.40	0.30	0.26	0.32
F1 Score (micro)	0.47	0.44	0.46	0.28	0.42	0.32	0.35

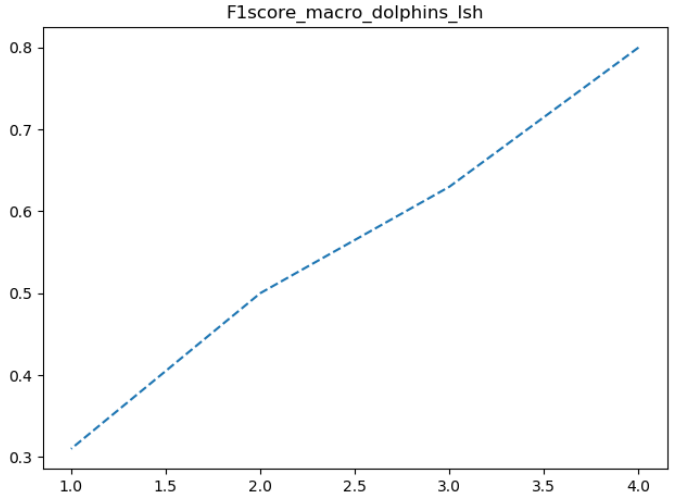
Accuracy (LSH) (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

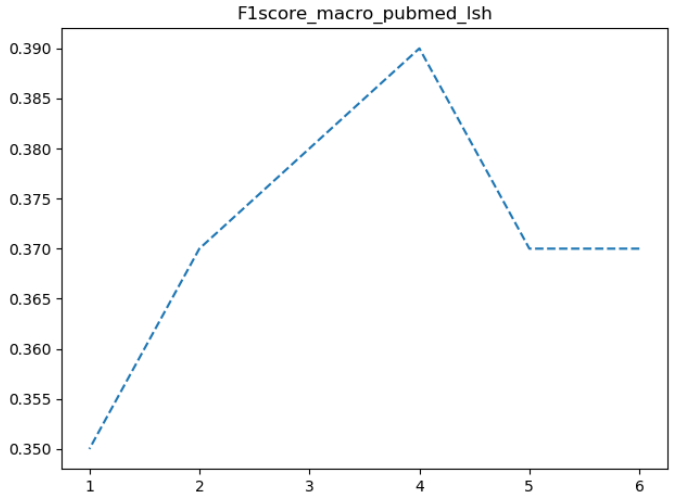


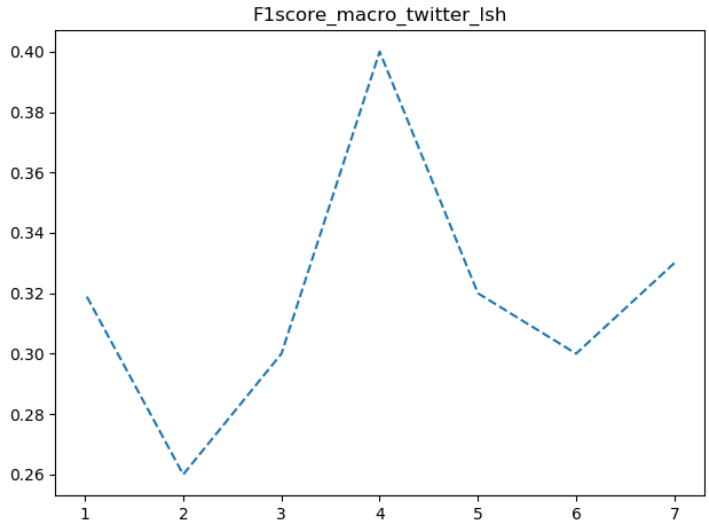




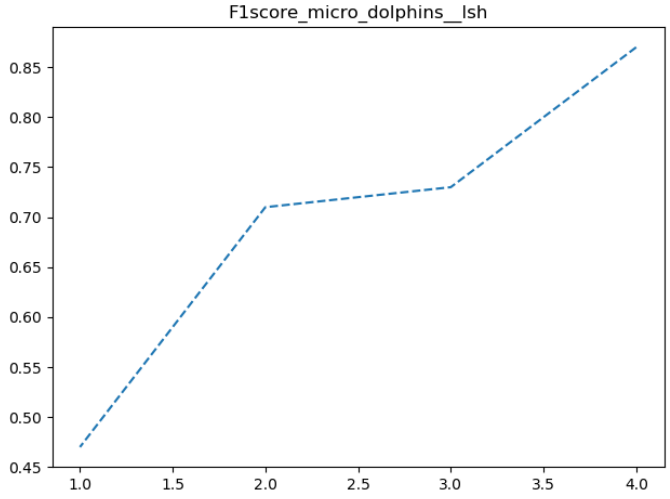
F1 Score macro: (LSH) (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)

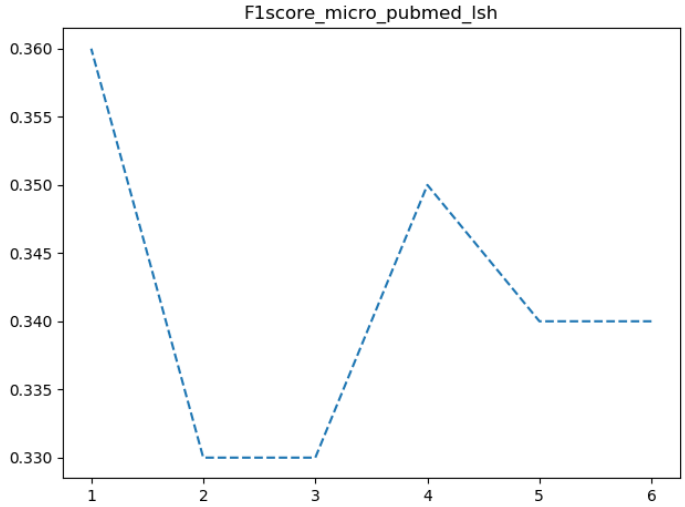


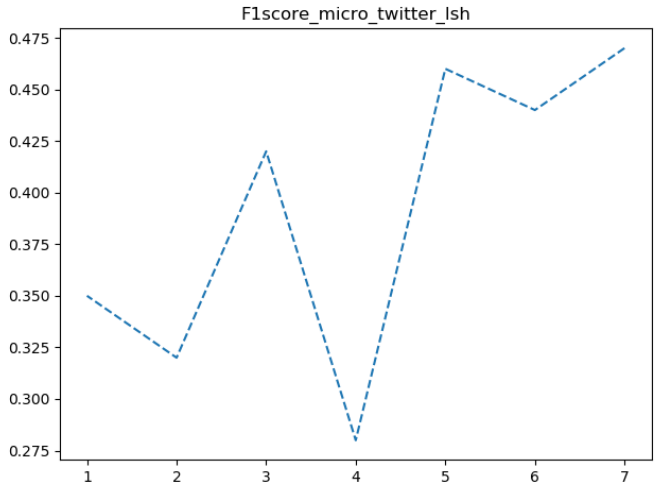




F1 Score (micro): (LSH) (Plots have x axis in $\log_2$ format. i.e 2 => 1, 32 => 5, etc)







Accuracy, F1 Score (PCA):

Dolphins**:**

	D= 16	8	4	2
Accuracy	0.90	0.90	0.90	0.90
F1 Score (macro)	0.87	0.86	0.86	0.86
F1 Score (micro)	0.9	0.9	0.9	0.9

Pubmed:

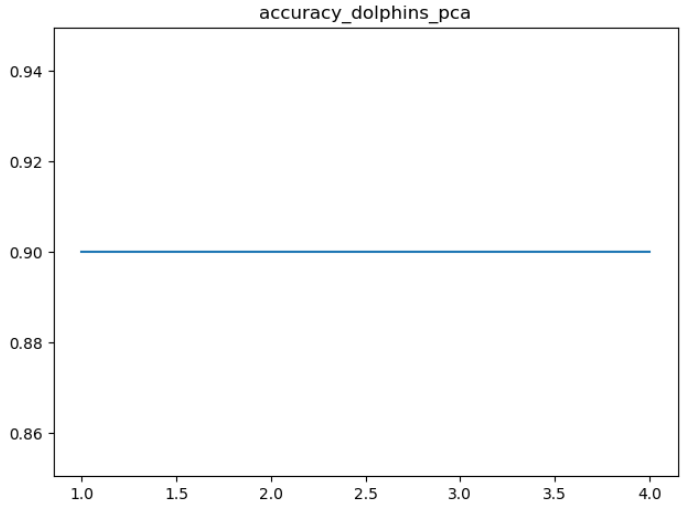
	D = 64	32	16	8	4	2
Accuracy	0.4296	0.431	0.4354	0.4384	0.436	0.4363
F1 Score (macro)	0.38	0.37	0.37	0.36	0.35	0.35
F1 Score (micro)	0.43	0.43	0.44	0.44	0.44	0.44

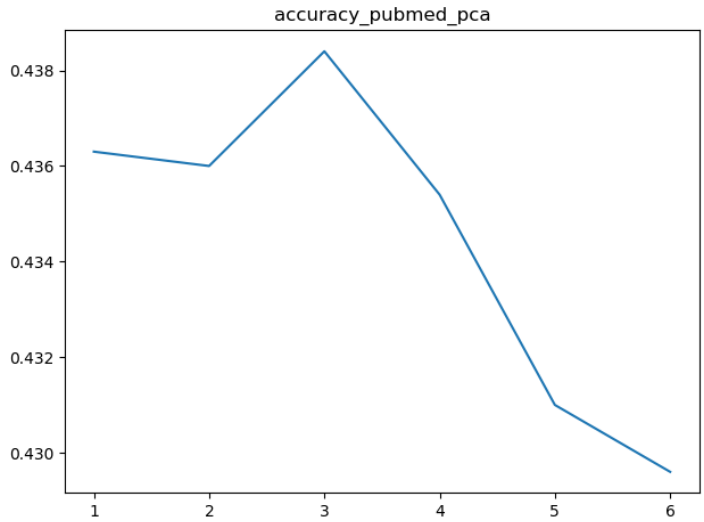
Twitter:

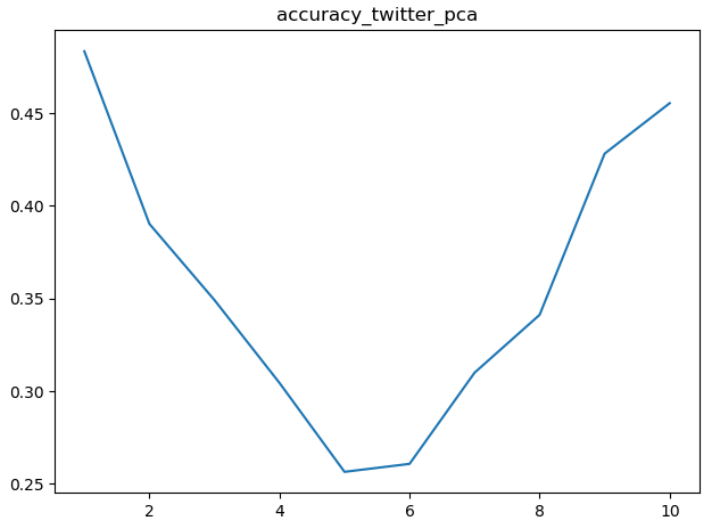
	D = 1024	512	256	128	64
Accuracy	0.4554	0.4282	0.3412	0.31	0.2608
F1 Score (macro)	0.37	0.36	0.32	0.29	0.24
F1 Score (micro)	0.45	0.43	0.34	0.30	0.26

	32	16	8	4	2
Accuracy	0.2565	0.3045	0.3492	0.3903	0.4834
F1 Score (macro)	0.25	0.25	0.25	0.264	0.31
F1 Score (micro)	0.26	0.30	0.35	0.39	0.48

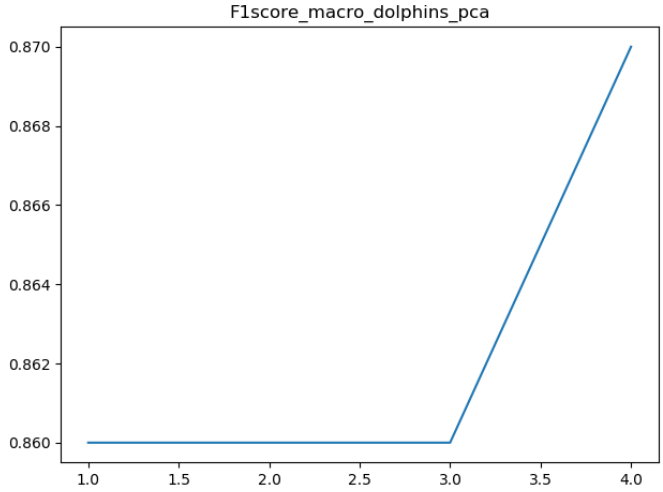
Accuracy (PCA):

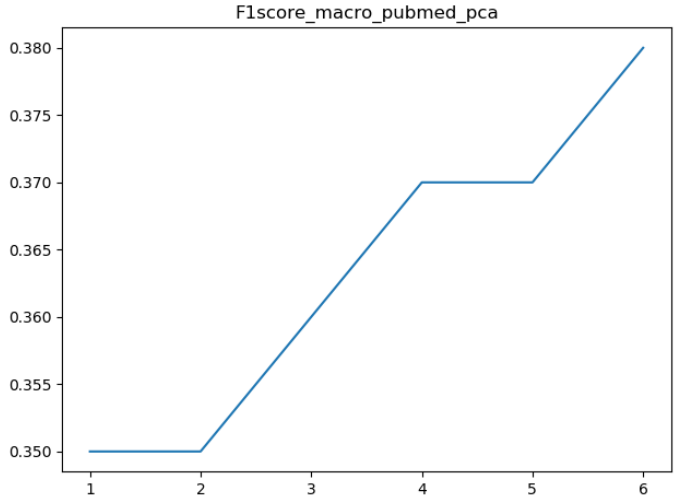


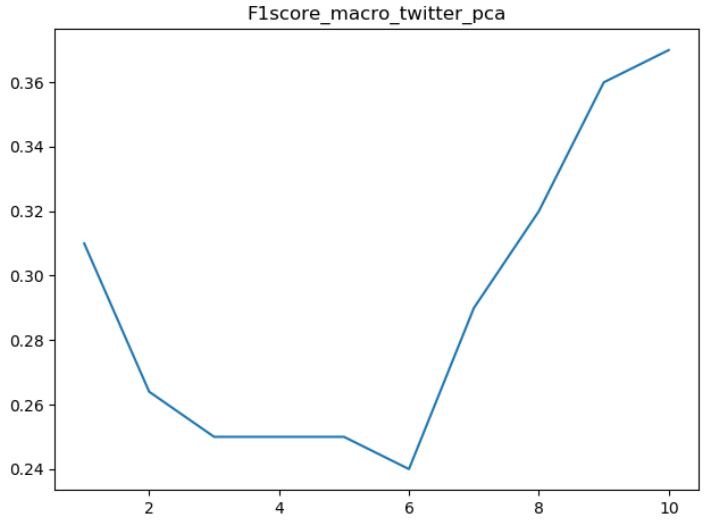




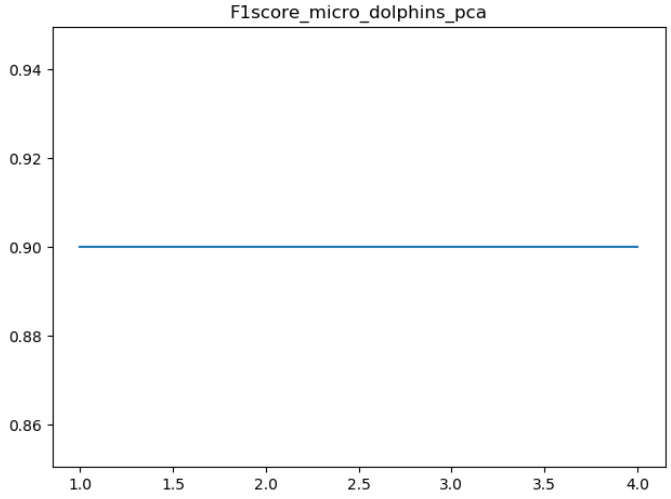
F1 Score (macro):

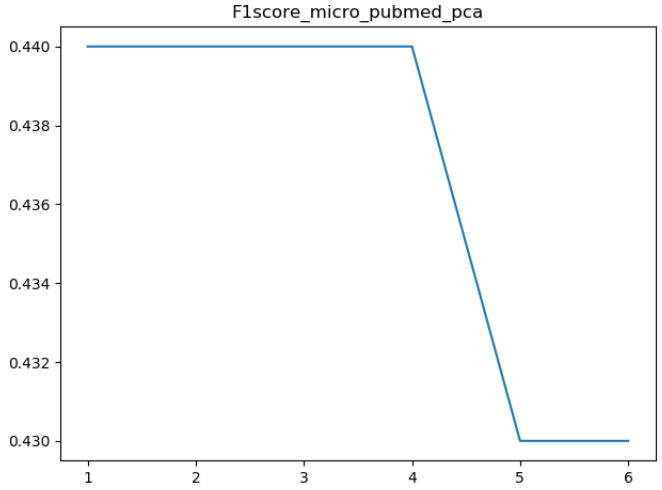


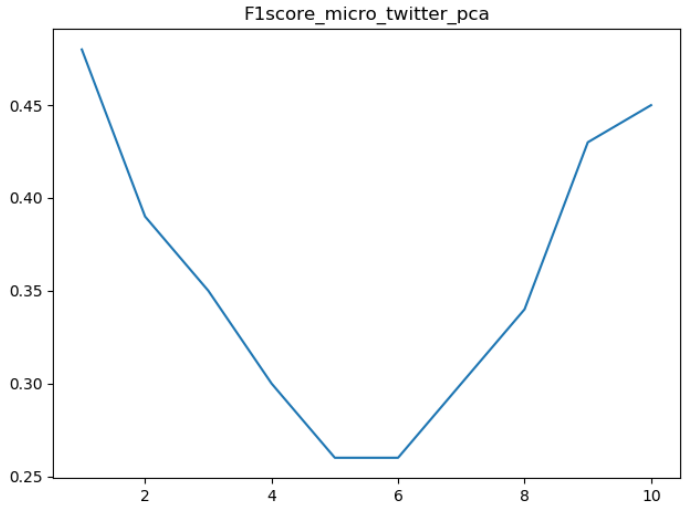




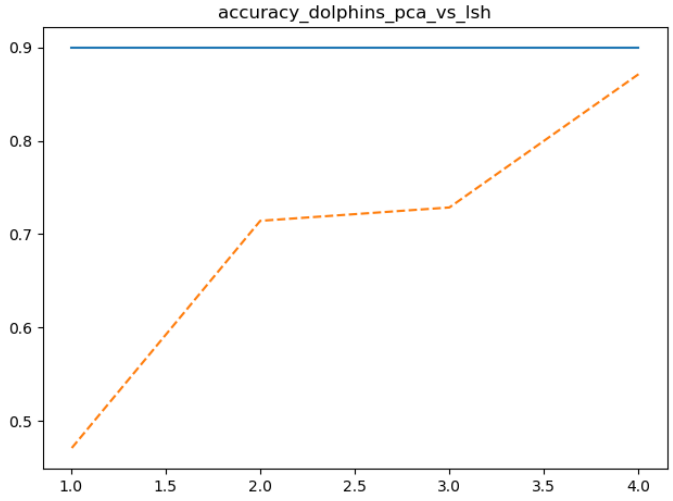
F1 Score (micro):

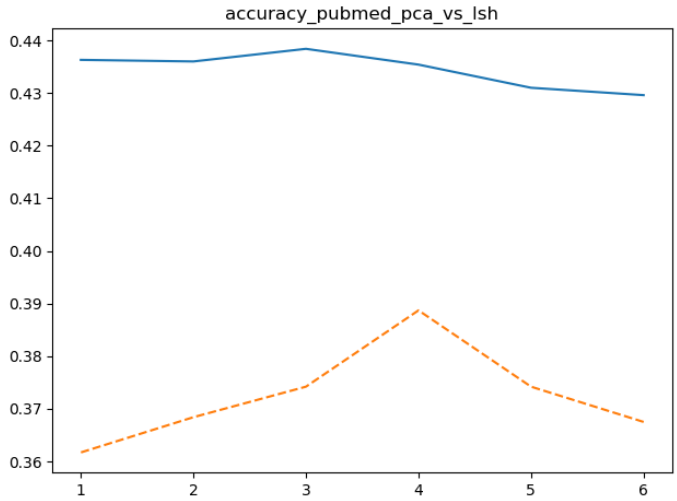


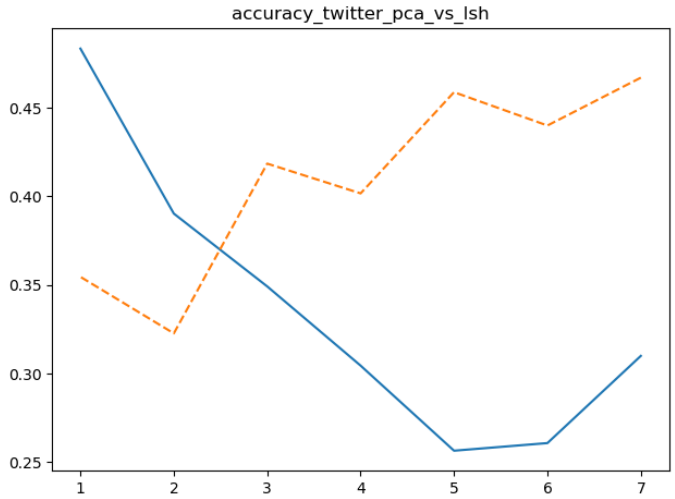




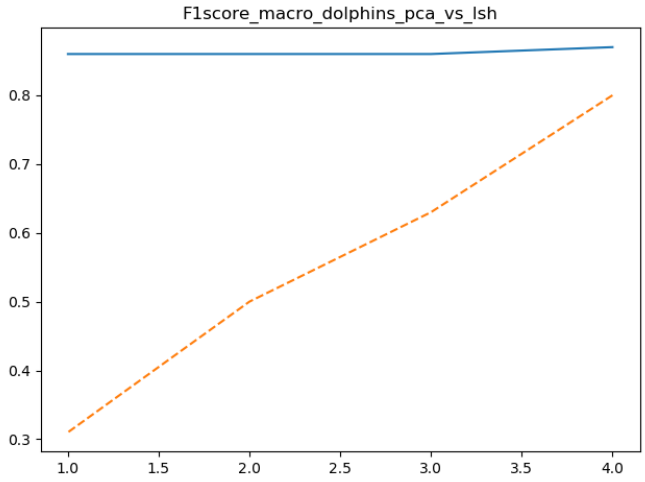
Comparison: Accuracy (Dotted line => LSH, Normal line => PCA)

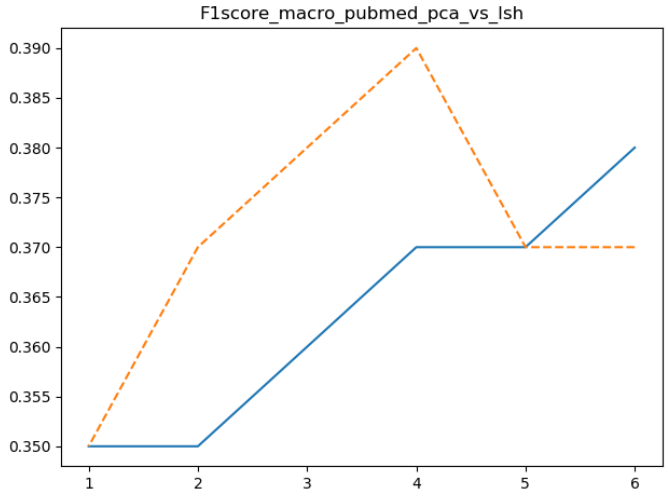


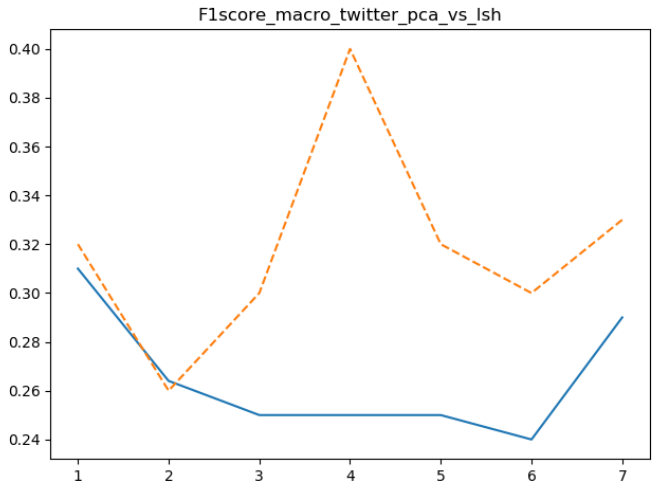




Comparison: F1 Score (macro):







Comparison: F1 Score (micro):