REPORT

Q1 : NAIVE BAYES

Part (a) :

- Implementing Naive Bayes by the Multinomial Event Model I obtain the following accuracies for the training and testing data files.

TRAINING DATA	:	**96.9553%** Accuracy
TESTING DATA	:	**95.0206%** Accuracy

- Used Laplace Smoothing and took log to avoid underflows.
- The size of the vocabulary obtained using both of the datas is 23585.

Part (b) :

- Random Prediction : Since there are 8 classes. We will randomly predict one of those 8. The actual class can also be one of the 8 so there are 8*8 = 64 total outcomes and just 8 cases when our prediction is same as the actual class. Hence favourable outcomes are 8. Hence the probability is 0.125 or 1/8 and hence **12.5%** will be the accuracy.
- Majority Prediction :Most of the time occurring class is ‘earn’ class which occurs 1083 times in the test set and total documents in the test set are 2189. So the accurate predictions will be just 1083 out of 2189. Hence the accuracy will be **49.47%.**
- We have a lot of improvement over the random and majority baselines. 45.5 increase in percentage from the majority baseline and 82.5 increase from the random baseline.

Part (c) :

- Confusion matrix of 8*8 size for test data is :

689	1	2	0	0	1	0	3
0	118	0	0	0	0	0	3
24	1	1056	0	0	0	0	2
0	3	1	3	0	0	0	3
0	0	0	0	49	24	0	8
0	0	2	0	0	79	0	6
3	10	0	0	0	0	14	9
0	0	2	0	0	1	0	72

- We have the 3rd entry as the highest in the diagonal entries. This means that 3rd class (‘earn’) was highest no. of times correctly predicted. Basically the (i,j)th entry in the matrix means, how many times the class was actually i and we predicted it to be nth class.
- Largest entry in the non diagonal ones are 24 which are bw the classes ‘earn’ and ‘acq’ and also bw the classes ‘interest’ and ‘money-fx’. These are the 2 pairs of classes which have most confusion bw them.
- The first pair is earn and acq which have 2 highest number of articles. But the second pair have very less number of articles as compared to the first one. Hence I observe that there is higher percentage or more sense of confusion bw the second pair than the first one. So basically same type of words are essentially used in those 2 pair of classes.

Part (d) :

- Confusion matrix of 8*8 size for test data after stop word removal and stemming is

687	1	4	0	0	1	0	3
2	116	0	0	0	0	0	3
24	1	1056	0	0	0	0	2
0	2	0	5	0	0	0	3
0	0	1	0	54	20	0	6
0	0	1	0	1	81	0	4
3	5	0	0	0	0	22	6
0	1	1	0	0	1	0	72

- The accuracy obtained in this case is **95.6144 %.** There is a minute increase in the accuracy. previously it was 95.02 %. Obviously since the stop words don't really differentiate bw any 2 classes. They may occur by chance higher number of times in some particular class documents then our model will learn that feature, but generally for a more huge data set the stop words won't behave in the same way. Hence they are the reason of noise in data so removal will help us improving its accuracy. Stemming will help in the same way as when the similar words will be treated in different ways then it will add on to the noise.
- The confusion matrix still gives the diagonal entry highest to the 'earn' class. But the highest non diagonal entry now becomes just one which goes to the poor 'earn and 'acq'. Removing the noise due to stop words and stemming contribute to decrease in the some of the values in the non diagonal entries. And also some of the diagonal entries have increased due to increases in accuracy.

Q2 : **SVM**

Part (a) :

- Comparing with the SVM Dual Objective { ∑**α**i - (0.5) ∑∑ (**α**i * **α**j * yi * yj * xi' * xj) } with the form mentioned in the question, I obtain b as a column vector of size m*1 and all ones. Q(i,j) is all the things in the double summation above except the **α**i * **α**j, and also the -0.5 included in the Q.
- After getting alpha from the CVX package after optimisation, the alpha values with
- The support vectors are in the separate file SV1.txt

Part (b) :

- Calculated w and b using the alphas obtained above W = ∑**α**i * yi * xi . And since its a noisy data first found such alpha which is >0 and <C and then for the index of that **α**, equated w' * xi + b = 1, to find the value of b = -1.833.
- Then calculated the accuracy by classifying into different classes as **61.667 %.**

Part (c) :

- The places where xi' * xj is present in term of Dual Objective and the places where we have to make prediction by w' * xi + b, we replace all those by the kernel function given to us.
- Accuracy obtained is **67.5 %.**
- We are not able to calculate w here directly but all the terms for w' * x can be calculated as the x'*x term is obtained hence can be replaced by the kernel function.
- We can find b by the same method we used in part(b). b = -6.11

- Accuracy obtained is higher in the gaussian case probably due to the fact that the data is distributed in there test data such that linear doesn't fit it as well the gaussian kernel does.
- The support vectors are stored in SV2.txt

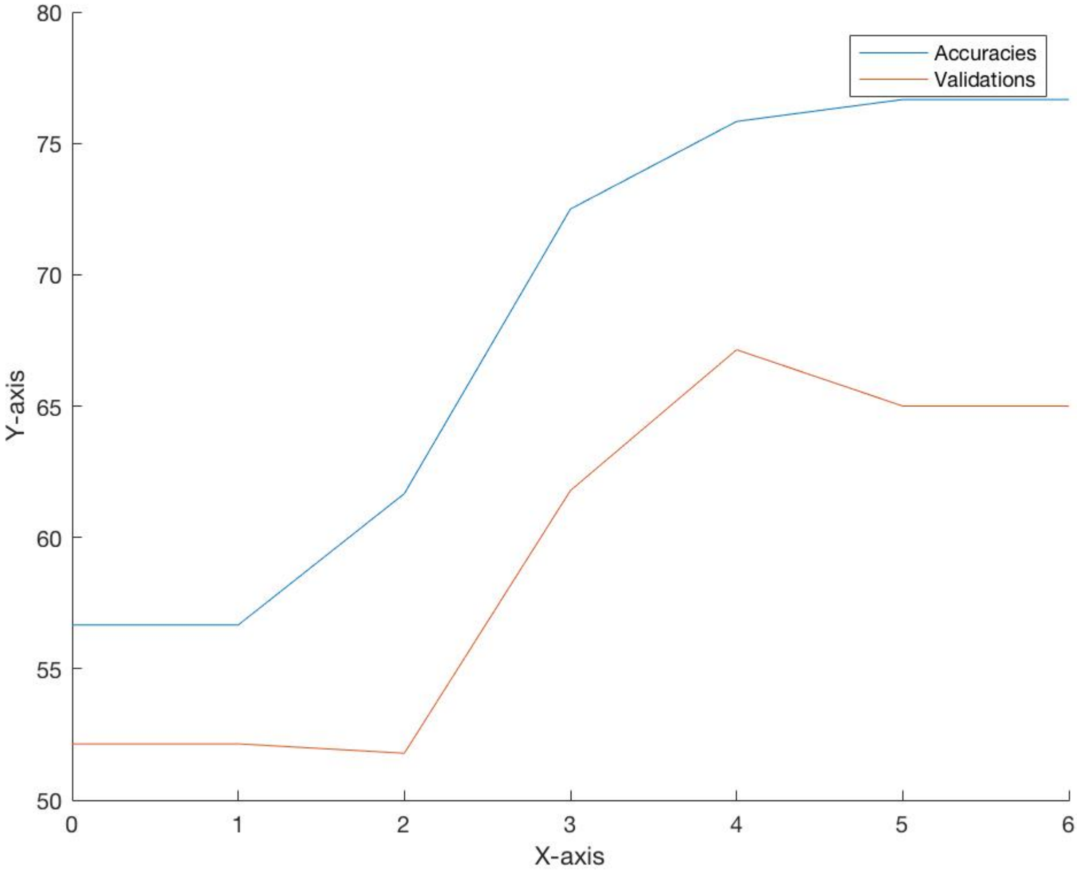
Part (d) :

- The accuracies obtained are 61.667 % and 67.5 % in the linear and gaussian kernel respectively for the LIBSVM library. These are exactly same as we got from the CVX package. SVM theory used in the both of the package is same hence we get the same accuracies. The accuracy will rather depend on the training data.
- Support vectors are in SV3.txt and SV4.txt respectively for linear and gaussian kernel.

Part (e) :

- Cross Validation Accuracy = 65%
- C = [1 10 100 1000 10000 100000 1000000]
- **Average Test Set Accuracies :**
- 56.6667 56.6667 61.6667 72.5000 75.8333 76.6667 76.6667
- **Average Validation Set Accuracies :**
- 52.1429 52.1429 51.7857 61.7857 67.1429 65.0000 65.0000

- In the plot below and the values above below the test set accuracies are higher than the validation set accuracies at each value of C. They peak at the highest value of C for the test set accuracies but they peak for validation accuracies little bit earlier. This happens because in validation the data is differntiated into different sets and each part of the data gets chance to be the one in training and in testing. Higher C make SVM prob closer to less noise.


Accuracies
Validations
Y-axis
X-axis
80
75
70
65
60
55
50
0
1
2
3
4
5
6