Note for code execution : The shell scripts are according to the submission instructions. But since there were dependencies in questions, like confusion matrix comparison etc.. I have combined below mentioned questions in one file.

1. Question 1 - a,b and c (calling any one of them will execute all 3)
2. Question 2 - Part B (iii) i.e. c of multi_class (confusion matrix are included in a and b, also added in the write up file below)

PART A

(a) Naive Bayes Implementation

(i) Steps Used for pre processing:

(1) Resolving short forms involving “not” i.e. haven’t -> have not, isn’t -> is not.

(2) Breaking other short forms like I’m to I m (separate by space).

(3) Remove non alpha characters except [“.” (to identify new sentence) , “ “ (space)]

(b) Accuracies:

(i) Training accuracy : 52.21 %

(ii) Test accuracy: 65.7 %

(iii) Random guess accuracy: 20.09 %

(iv) Most frequent class accuracy: 66.08 %

(c) Confusion Matrix

Predicted **Actual**	1	2	3	4	5
1	3	1	2	2	9
2	0	0	1	1	1
3	1	1	2	5	7
4	1	5	12	29	61
5	223	319	1069	3071	9174

- Class 5 has the highest diagonal entry, which means class 5 has the highest number of true classifications (True positives).
- Class 2 has no correct predictions.
- It seems that the matrix has this nature because of biased data.
- Class wise F1 score [2.4, 0, 0.4, 1.8, 79.4]
- Average = 16.8

(d) Steps performed and Observations:

(i) Stemmed tokenised words using “Snowball stemmer”, was giving better result than porter stemmer.

(ii) Few stopwords - not, like etc.. are important keywords for review, so instead of using direct nltk stopwords, I analysed distribution of tokenized words and selected most frequent words (which have intersection with stop words) and also occur in every class covering a good amount of proportion.

(iii) Test Accuracy : 63.05 %

(iv) Accuracy is reduced than part “a”, the reason seems that due to biased data, term frequencies of tokens in class 5 is high which increases its probability and since even in test data this occurs for most of the time, so even a randomness/less confidence in prediction will output class 5. Reducing common words reduces the probability of class 5 being predicted as target and hence accuracy decreases. Even though accuracy decreases, F1 scores are observed to increase.

Predicted **Actual**	1	2	3	4	5
1	13	8	18	54	196
2	2	0	5	15	41
3	0	0	1	5	25
4	9	2	18	65	244
5	204	316	1044	2969	8746

(v) We can see here that for other classes True positives have increased.

(vi) Class wise F1 score = [5.0, 0, 0.2, 3.8, 77.6] (Now)

(vii) Average : 17.32 **(Increased)**

(e) Feature engineering

(i) Using Bi grams:

  (1) Test accuracy : 61.08 % (It reduces by some amount).

  (2) Training accuracy : 89%

  (3) Also it was observed here that the model was majorly predicting the “majority” class i.e. Class 5.

  (4) So the F1 score has decreased. Since bi grams are very less frequent laplace smoothing penalise frequencies and smaller frequencies are almost diminished(the less frequent classes)

(ii) Using Tri grams:

  (1) Observed to be worse than Bi-grams, because of the same reason mentioned above.

(iii) Syntactic features

  (1) Here we can exploit features like - sentence length, number of unique words, ratio of unique words to total words in sentences, use of special characters (@,*,# - which might be more frequent in negative reviews)

  (2) Data representation using this, might not help to generalise better on unseen data.

(iv) Ensemble technique where : unigram, and syntactic features can be used with weighted sum.

(f) Using Summary Text

(i) Summary field is observed to be a condensed form of reviewText field.

(ii) Using only summary field keeps accuracy almost same i.e.: 66.5 %, but the average F1 score over all classes is observed to be improved.

(iii) One better way can be to take weighted average of predicted probability to compute target class, as it is observed that for some sentences summary works better while for some other unigrams work better.

PART B

A) **BINARY CLASSIFICATION SVM**
   a) Linear Kernel using CVOPT
      i) Training Time: 157 secs
      ii) # support vectors: 209
      iii) Accuracy: 96.47%
   b) Gaussian Kernel using CVXOPT
      i) Training Time: 50 secs
      ii) # support vectors: 1468
      iii) Accuracy: 98.4%
   c) LIBSVM
      i) Linear Kernel:
         (1) Training Time: 2secs
         (2) # support vectors: 208
         (3) Accuracy:96.97%
      ii) Gaussian Kernel:
         (1) Training Time: 2secs
         (2) # support vectors:1463
         (3) Accuracy:98.43
   - Near same accuracies for both the classes, but the execution time varies a lot.
   - Also for self implemented part threshold for support vectors was iteratively checked and value for best accuracy was selected.

B) MULTI CLASS CLASSIFICATION SVM

a) Multi class SVM using CVXOPT (kC2 classifiers)

i) Test Accuracy : 97.23 %

ii) Time : 1246 seconds

b) Multi class SVM using LIBSVM (kC2 classifiers)

i) Test Accuracy: 97.24%

ii) Time: 492 secs

c) Confusion matrix:

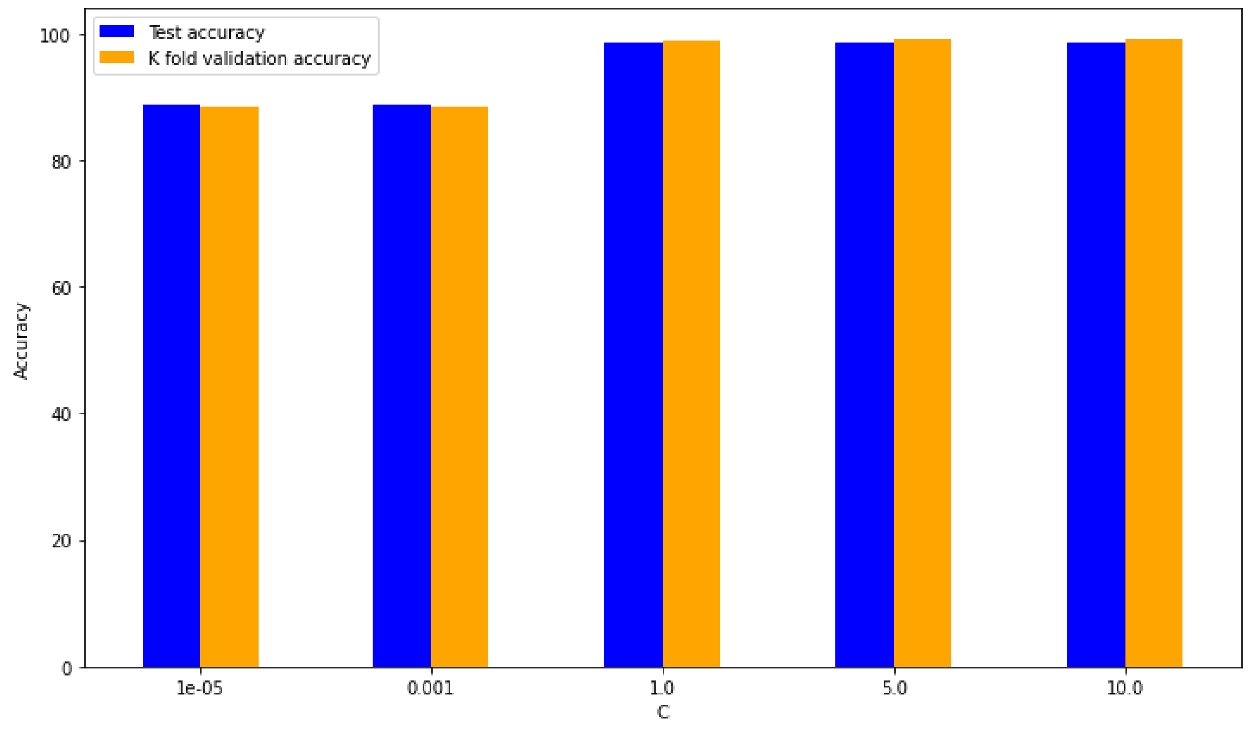
Surprisingly the confusion matrix for both cases is the same (reason being same accuracy in part a and b).

	PREDICTED VALUES										
ACTUAL VALUES		0	1	2	3	4	5	6	7	8	9
	0	969	0	1	0	0	3	4	1	2	0
	1	0	1122	3	2	0	2	2	0	3	1
	2	4	0	1000	4	2	0	1	6	**15**	0
	3	0	0	8	984	0	4	0	6	5	3
	4	0	0	4	0	962	0	6	0	2	8
	5	2	0	3	6	1	866	7	1	5	1
	6	6	3	0	0	4	4	939	0	2	0
	7	1	4	**19**	2	4	0	0	987	2	9
	8	4	0	3	**10**	3	5	1	3	942	3
	9	5	4	3	8	**13**	3	0	9	**12**	952

1. Here the accuracies are same because the number of support vectors for CVXOPT were selected by iterative check method to attain best accuracy. So the results are same for both the approaches.
2. Highest miss classification : Actual : 7 , Predicted : 2. Few such sample were observed.

d) K fold cross validation with LIBSVM (Binary classification)

C	5 fold cross validation accuracy	Test accuracy
0.00001	88.5	57.8
0.001	88.5	96.8
1	98.65	**99.14**
5	98.75	**99.14**
10	**98.87**	**99.14**



C= 10 gives best validation accuracy, but test accuracy results are same for C =1,5 and 10. It seems that due to validation check, the model stops before it starts getting overfitted.