Question 1: Text classification using naive bayes

a) The naive bayes algorithm was implemented as follows:

1. The vocabulary was obtained by considering all words present in the training set. Each word was replaced by a corresponding integer
2. The parameters θ and ϕ were estimated in the **log scale** according to
   $\theta_{l/k} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=k\} \sum_{j=1}^{n^{(i)}} 1\{x_j{}^{(i)}=l\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)}=k\} \cdot n^{(i)} + |V|}$
   $\phi_k = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=k\}}{m}$
3. For classifying an example, the best class was chosen according to $argmax_k \quad \phi_k \cdot \Pi_{j=1}^{n} \theta_{l=x_j/k}$ This was again calculated in log scale to avoid underflow

Vocabulary size obtained: 855769
The prediction accuracies obtained are
Training set: 69.61%
Test set: 61.87%
The cumulative time taken for training and testing was 282.79s

b) The accuracies obtained are as follows:
Random prediction (test set) : 19.95%
Majority prediction (test set): 43.99%
The naive bayes model provides an improvement of classifying 41.91% more examples correctly compared to random prediction, and 17.88% more examples correctly over majority prediction.

c) Confusion matrix obtained for naive bayes model over the test set is as follows:
Here columns represent the actual classes 1-5, and rows represent the predicted classes 1-5

$$\begin{bmatrix} 15493 & 3377 & 1615 & 1067 & 2479 \\ 2174 & 1962 & 819 & 266 & 107 \\ 1124 & 2935 & 3646 & 1137 & 253 \\ 873 & 2086 & 7302 & 18545 & 12897 \\ 505 & 478 & 1149 & 8343 & 43086 \end{bmatrix}$$

Here class 5 (corresponding to 5 star reviews) has the largest diagonal entry of 43086 correct predictions (73.2% of total 5 star reviews). The largest % of correctly classified reviews belong to class 1 (15493, 76.8% of 1 star reviews). This shows that the model performs best on the most positive and most negative reviews. This can be because our model uses words as features, and both 5 and 1 star reviews would have a high concentration of positive and

negative words respectively.
We also observe that in case of misclassifications, the predicted class is generally near to the actual class (off by 1-2 stars)

d) On performing stopword removal and stemming according to the function provided (and using lru_cache to speed up), the performance of the new model is as follows:
Vocabulary size: 270956
Accuracy (train set): 64.53%
Accuracy (test set): 60.68%
Time taken (training + testing): 911.6s

We observe that the vocabulary size decreases. However the accuracies decrease as compared to the earlier model, suggesting that in case of the given dataset stemming and stopword removal are actually degrading the performance.

e) Modifications via feature engineering which were tried are as follows:

a. **Bigram**
Words are replaced by bi-grams (the word followed by its next word) in the dataset. The model was trained and tested with and without performing stemming:
Without stemming or stopword removal:
Vocabulary size: 8927743
Accuracy (train set): 87.26%
Accuracy (test set): 63.58%
Time taken (training+testing): 423.62s

With stemming and stopword removal:
Vocabulary size: 5407689
Accuracy (train set): 86.14%
Accuracy (test set): 63.92%
Time taken (training+testing): 1110.79s

Thus we observe that both models perform better than our original model of part (a) on both train and test sets, but have higher memory and time requirements. The first model has a huge increase in vocabulary size compared to the original model, and provides a 1.7% accuracy increase on the test set. Compared to the first model, the second model sees a decrease in the vocabulary size due to stemming (but is still huge), with the test accuracy being 0.3% more

b. **Limiting features by considering only the 100,000 least occuring words**

In the training set, the 100000 least occuring words are identified and the rest are filtered out. The filtering is then also done on the test set according to the vocabulary obtained. This is because the least occuring words are likely to have a greater impact on the classification, compared to more frequently occuring words.
The performance of the model is as follows:

Without stemming or stopword removal:
Accuracy (train set): 47.33%
Accuracy (test set): 43.87%
Time taken (training+testing): 54.72s

With stemming and stopword removal:
Accuracy (train set): 48.11%
Accuracy (test set): 43.96%
Time taken (training+testing): 870.24s

The models have a poor accuracy compared to the original model in part (a), however they are faster in training and testing (due to the vocabulary being fixed at 100000 words)

c. **Inverse class frequency**
If $f_{l,k}$ denotes the count of word l in class k, then it was changed as $f_{l,k} \leftarrow f_{l,k} * \frac{5+1}{c_l+1}$, where c_l is the no. of classes in which word l appears
This gave more emphasis on the words which occured in less number of classes, compared to the words which were common across classes.
The naive bayes parameter θ then became
$\theta_{l/k} = \frac{f_{l,k}+1}{\sum_{l=1}^{|V|} f_{l,k} + |V|}$

The performance of the model is as follows:

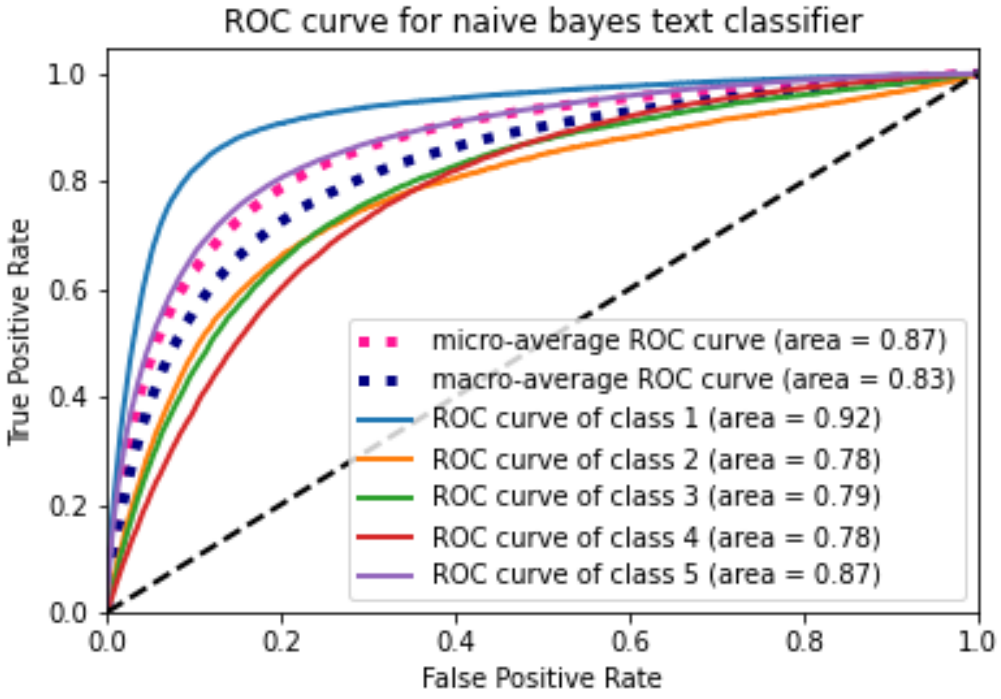
Without stemming or stopword removal
Accuracy (train set): 72.38%
Accuracy (test set): 61.76%
Time taken (training+testing): 266.23s

With stemming, stopword removal and using the bigram model
Accuracy (train set): 91.73%
Accuracy (test set): 63.77%
Time taken: 958.75s

Thus the model performs better than the original model of part (a) or (d).

f) The ROC curve (for micro and macro-average, as well as for individual classes) was plotted using scikit's implementation (at https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html)

We observe that the curve lies in the top left region, with large area under the curve. The curve is also steep. It implies that our model is performing well, since we have a large true positive rate and low false positive rate. Also, since curve for class 1 is at the most top left, the model performs best for class 1, which is what was observed in the confusion matrix as well.



Question 2: Fashion MNIST classification using SVM

a) **Binary classification**

i. The dual objective to be solved is as follows
$min_\alpha \quad -\sum_{i=1}^{m} \alpha_i + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)}x^{(j)}$
subject to $\quad 0 \leq \alpha_i \leq C \quad \forall i$
and $\quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$
The matrices for CVXOPT (P,q,G,h,A,b) were formed according to this problem.
The support vectors were identified as those for which corresponding value of α was greater than a threshold (taken to be 10^{-5})

w and b were then calculated using these support vectors and corresponding α values.

The model's performance on data corresponding to classes 5 and 6 is as follows:
86 support vectors found
Training time: 153.02s
Train set accuracy: 100.0%
Validation set accuracy: 99.8%
Test set accuracy: 99.6%

We observe a very high accuracy because the 2 classes in consideration(sandal and shirt) are visually very different.

ii. For gaussian kernel, the matrix P had to be changed according to the new objective:
$min_\alpha \quad -\sum_{i=1}^{m} \alpha_i + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y^{(i)}y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)})$
subject to $\quad 0 \leq \alpha_i \leq C \quad \forall i$
and $\quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

The model's performance on data corresponding to classes 5 and 6 is as follows:
1130 support vectors found
Training time: 101.50s
Train set accuracy: 100.0%
Validation set accuracy: 99.6%
Test set accuracy: 99.8%

b) **Multi-class classification**

i. The classifier was implemented by training 45 binary classifiers. At the time of prediction, the most votes of all classifiers was declared the chosen class. In case of a tie, the score of the class (sum of $w^T \cdot x$ values where the class earned a vote) was considered.

The performance of the model is as follows:
Training time: 4130.15s
Validation set accuracy: 84.84%
Test set accuracy: 84.92%

ii. The performance of scikit's classifier (using sklearn.svm.SVC) with C=1, kernel as rbf and gamma=0.05 is as follows:
Training time: 353.71s
Validation set accuracy: 87.92%

Test set accuracy: 88.08%

Thus the performance of scikit's classifier is better in terms of both the training time, as well as the accuracies obtained

iii. The confusion matrices obtained are as follows:

For multi-class classification
In these the columns represent the actual classes 1-10 and the rows represent the predicted classes 1-10

```
Confusion matrix (Validation set):
[[202   0   2  12   0   0  20   0   0   0]
 [  2 240   0   9   2   0   0   0   0   0]
 [  1   2 208   0  30   0  27   0   1   0]
 [  4   2   1 195   5   1   1   0   1   0]
 [  0   0  13   6 185   0  11   0   0   0]
 [  0   0   0   0   0 227   0  26   0   2]
 [ 36   3  14  19  19   0 182   0   1   0]
 [  0   0   0   0   0   1   0 198   2   4]
 [  5   3  12   9   9  15   9   4 245   5]
 [  0   0   0   0   0   6   0  22   0 239]]

Confusion matrix (Test set):
[[403   0   0  17   0   1  56   0   1   0]
 [  1 484   0  10   1   0   1   0   0   0]
 [  7   6 414   2  54   0  55   0   1   0]
 [  7   2   4 411  13   0   6   0   0   0]
 [  0   0  26   6 368   0  21   0   0   0]
 [  0   0   0   0   0 434   0  49   0   5]
 [ 62   6  42  36  50   0 340   0   3   0]
 [  0   0   0   0   0   7   0 411   0   6]
 [ 20   2  14  18  14  46  21   6 495   3]
 [  0   0   0   0   0  12   0  34   0 486]]
```

Figure 1: **Confusion matrices for 2(b).i.(CVXOPT multi-class classifier)**

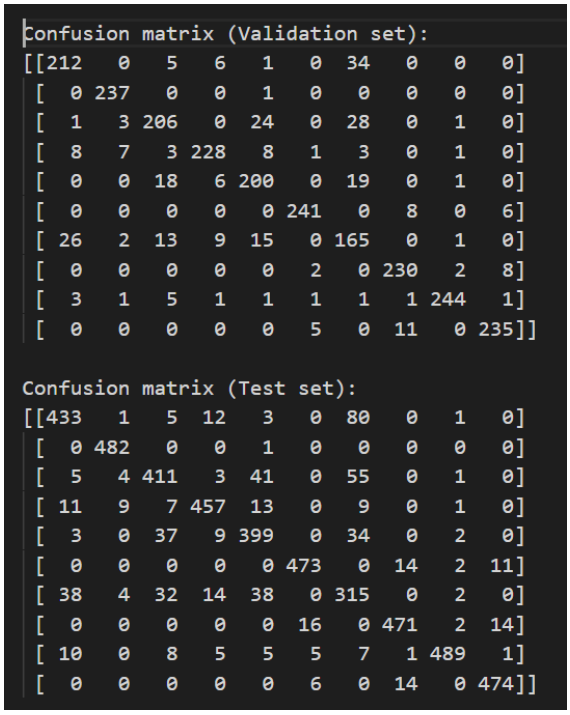


Figure 2: **Confusion matrices for 2(b).ii.(Scikit's multiclass classifier)**

Table 1: Significant misclassifications (> 20 examples in any matrix) for various classes

Actual class	Most wrongly classified as
0 (Tshirt/top)	6
1 (Trouser)	-
2 (Pullover)	4, 6
3 (Dress)	6
4 (Coat)	2, 6
5 (Sandal)	8
6 (shirt)	0,2,4,8
7 (sneaker)	5,9
8 (Bag)	-
9 (ankle boot)	-

It is observed that the mis-classifications occur with a similar clothing item class.
Items like trousers, bag and ankle boot (which are generally unique looking than other classes) are generally classified the best. Other items are often misclassified into a visually similar item (like tshirt as shirt, sneaker as sandal, coat as pullover). So the results make sense.

The above trends are observed among all 4 matrices in general. However, scikit's classifier in general fares a lot better in some cases (e.g. class 7) thus having an overall greater accuracy.

For binary classification

Here columns represent the actual classes 5-6, and rows represent the predicted classes 5-6.

Confusion matrices for 2(a).i.(Linear kernel SVM)

Validation set: $\begin{bmatrix} 249 & 0 \\ 1 & 250 \end{bmatrix}$

Test set: $\begin{bmatrix} 496 & 0 \\ 4 & 500 \end{bmatrix}$

Confusion matrices for 2(a).ii.(Gaussian kernel SVM)

Validation set: $\begin{bmatrix} 248 & 0 \\ 2 & 250 \end{bmatrix}$

Test set: $\begin{bmatrix} 498 & 0 \\ 2 & 500 \end{bmatrix}$

We observe a very high accuracy. All examples of class 6 (shirt) are classified correctly, whereas very few examples of class 5 (sandal) are misclassified.

iv. The accuracies obtained are as follows. 2 graphs are shown, one with all 5 values, and one with C=1,5,10

Table 2: Accuracies for different value of parameter C

value of C	5-fold cross validation accuracy (%)	test set accuracy (%)
10^{-5}	9.46	57.36
10^{-3}	9.46	57.36
1	87.85	88.08
5	88.32	88.28
10	88.32	88.24

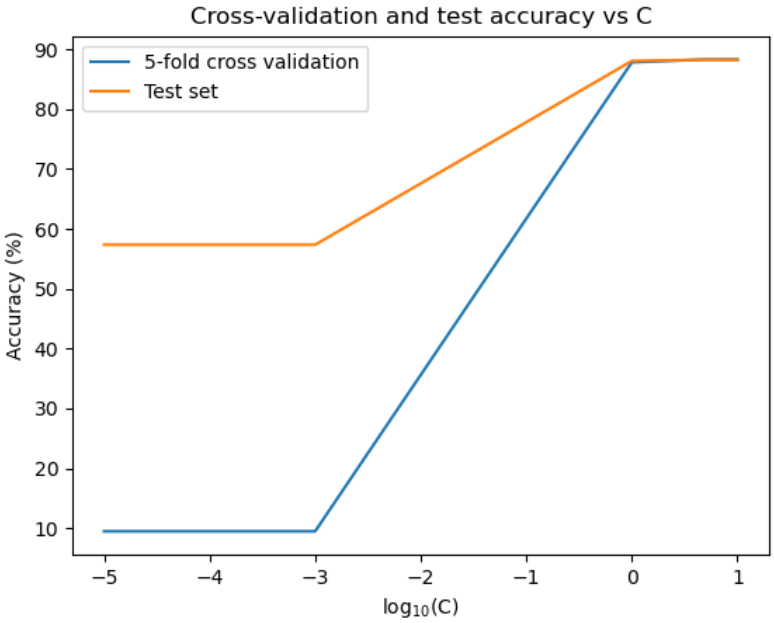


Figure 3: For all values of C

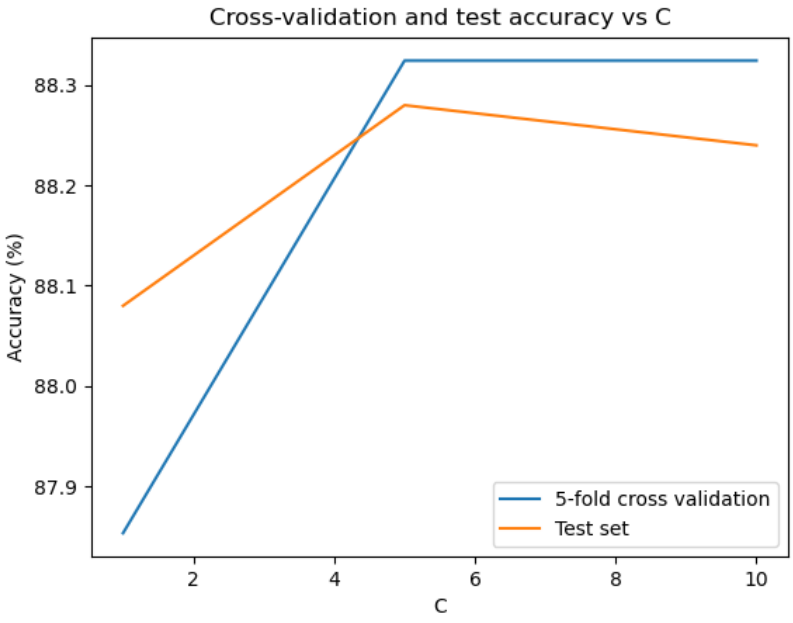


Figure 4: For C=1,5,10

Thus it is observed that C=5 and C=10 perform best in 5-fold cross validation. C=5 gives the best accuracy on the test set. C=10^{-3} and

10^{-5} perform poorly on both the sets. Thus we conclude that C=5 is the best value to be selected.

Question 3: Large scale text classification - Naive bayes v/s SVM (Liblinear) v/s SVM (SGD)

CountVectorizer was used to build a sparse matrix corresponding to the given data, containing count of each word in the vocabulary for each class. This matrix was used for the naive bayes classifier.
TfidfTransformer was used to transform the above into Tf-idf vectors, which were then used for the two SVM algorithms. This is because LinearSVC was not converging when the count matrix was used, but converged when Tf-idf vectors were used.

Naive bayes was implemented using scikit's MultinomialNB with default parameters.
SVM (LibLinear) was implemented using scikit's LinearSVC with tol=10^{-5}, and C=0.1 after tuning.
SVM (SGD) was implemented using scikit's SGDClassifier with alpha=1e-6 and max_iter=1000 after tuning.

Table 3: Tuning C in LinearSVC

C	Validation set accuracy (%)
0.001	59.99
0.1	67.57
0.5	67.09
1	66.65
2	65.99
5	64.91
10	64.01

Table 4: Tuning alpha, max_iter in SGDClassifier

alpha	max_iter	Validation set accuracy (%)
0.0001	500	63.12
0.0001	1000	63.15
0.0001	2000	63.06
1e-5	500	66.12
1e-5	1000	65.68
1e-5	2000	65.87
1e-6	500	66.38
1e-6	1000	66.58
1e-6	2000	66.54

Table 5: Test set accuracy and training time for different algorithms (final tuned parameters)

Algorithm	Test set accuracy (%)	Training time (s)
Naive bayes	60.03	0.4
SVM (LibLinear)	67.78	35.68
SVM (SGD)	66.55	22.29

Thus naive bayes trains the quickest but also has lesser accuracy. SVM (Liblinear) reaches the maximum accuracy but takes more time to train. SGD fares similarly to liblinear but has slightly lesser accuracy and training time.