1 Text Classification

For a, b, c, f do `python3 model1.py <train_data> <test_data> <output_file>`

For d do `python3 model2.py <train_data> <test_data> <output_file>`

For e do `python3 model3.py <train_data> <test_data> <output_file>`

For autograding i.e best model do `python3 nbbest.py <train_data> <test_data> <output_file>` or `./run.sh 1 <train_data> <test_data> <output_file>`

(a) Implemented an event model that follows multionomial distribution over the words. Assumed Naive Bayes assumption which states that any two distinct words in the document are conditionally independent of each other given their class labels. Laplace smoothing has been done with C = 1 to avoid zero probabilities.

$$\text{Naive Bayes Assumption: P(x, y)} = \prod P(x_j|y) \cdot P(y)$$

$$\phi_k = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} == k\}}{m}$$

$$\theta_{j=l/k} = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} == k\} \cdot \sum_{j=1}^{n(l)} \mathbb{1}\{x_j^{(i)} == l\} + 1}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} == k\} \cdot n^{(l)} + |V|}$$

Observations:

1. `Number of words in vocabulary`: 186850
2. `Training Set Accuracy`: 63.11753092328631%
3. `Test Set Accuracy`: 60.15719648813174%

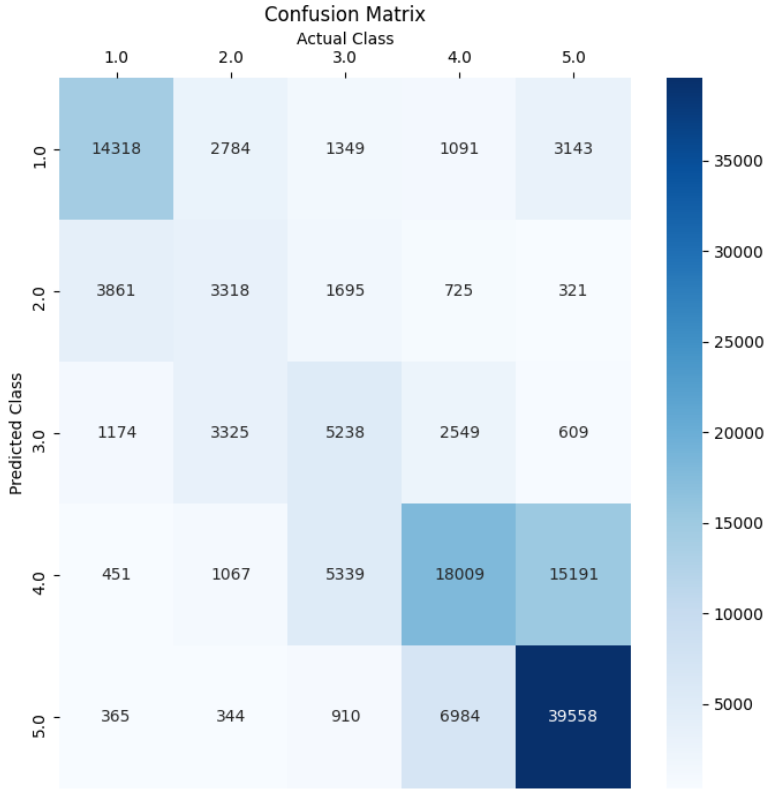
(b) Random Prediction and Majority Prediction

1. `Accuracy over Test Set (Random Prediction)`: 11.472651400708955%
2. `Accuracy over Test Set (Majority Prediction)`: 43.9895900327555%
3. `Improvement factor of the classifier over random baseline`: 5.244x
4. `Improvement factor of the classifier over majority baseline`: 1.368x

(c) Confusion Matrix

1. `True Positives`: 80441
2. `False Positives`: [20169, 10838, 14531, 29358, 58822]
3. `False Negatives`: [22685, 9920, 12895, 40057, 48161]

4. `Category with highest value of diagonal entry`: 39558 Reviews with 5 stars. This means that reviews with 5 stars are predicted best by the classifier.
5. **Conclusion**: The classifier is able to predict reviews with 1, 4 and 5 stars very well as they have very high true positive values The classifier is not able to distinguish reviews with 2 and 3 stars as can be seen from the confusion matrix that they all have roughly same values. Around 15191 reviews have been predicted to have 4 stars when the actual class is 5.



(d) Stemming and Stopwords Removal has been done using Porter Stemmer of nltk.

Observations:

1. `Number of words in vocabulary after stemming and stopwords removal`: 149710
2. `Training Set accuracy after stemming and stopwords removal`: 62.48784755979001%
3. `Test Set accuracy after stemming and stopwords removal`: 59.76158781914177%
4. Confusion Matrix:
   (a) `True Positives`: 79912
   (b) `False Positives`: [20169, 10838, 14531, 29358, 58822]
   (c) `False Negatives`: [23156, 9668, 12253, 38995, 49646]

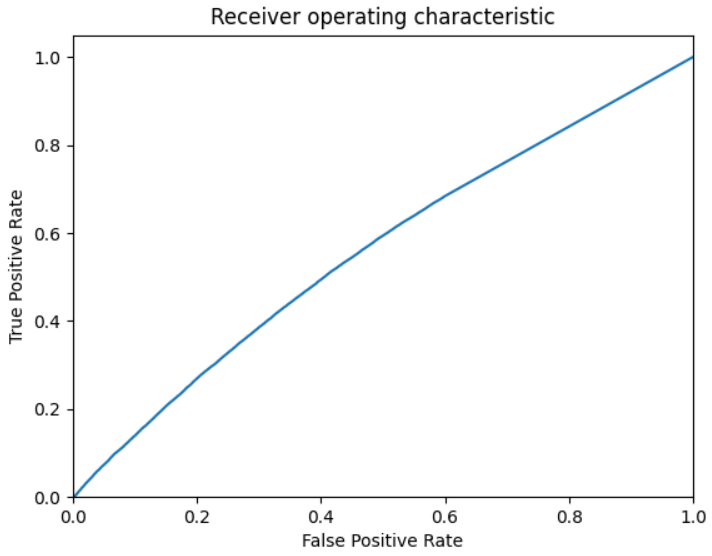
Conclusion: There is not an increase in test accuracy as compared to the model with no stemming and stopwords removal because of class imbalance.

(e) Feature Engineering - Two features added are bigrams and trigrams.

Features	**Unigrams + Bigrams (Stemmed Data)**	**Unigrams + Trigrams (Stemmed Data)**	**Unigrams + Bigrams + Trigrams (Stemmed Data)**
`Vocabulary Size`	2431992	3671660	5953942
`Training Accuracy`	61.21%	60.0%	58.9%
`Test Accuracy`	60.9%	59.95%	58.0%

Before adding features Test Accuracy on Stemmed Data is 59.76%. After adding Bigrams as a feature Test accuracy increases to 60.9%. After adding trigrams as a feature test accuracy increases to 59.95%. However adding both the features decreases the test accuracy to 58.0% On adding more and more features like length of word, higher grams and TF-IDF the accuracy started to decrease suggesting overfitting of the model on training data. Bigrams and Trigrams give a slight imporovement in the test accuracy as compared to model2.

(f) ROC Curve - ROC Curve has been done using micro-averaging. We can also draw different ROC Curves for each label.



Observation: The ROC curve turned out as expected. When the threshold is 1, then all examples are classified as 0. There are no positives, thus both the true positive and false positive rates are 1. Similarly, when threshold is 0, there are no negatives so both the rates are objectively 1.

2 Fashion MNIST Article Classification

For a(i) do `python3 BinaryLinear.py <train_data> <test_data>`

For a(ii) do `python3 BinaryGaussian.py <train_data> <test_data>`

For b(i), (iii) do `python3 MultiCVXOPT.py <train_data> <test_data> <output_file>`

For b(ii) do `python3 MultiScikit.py <train_data> <test_data>`

For b(iv) do `python3 kFoldSVM.py <train_data> <test_data>`

For autograding i.e best model do `python3 svmbest.py <train_data> <test_data> <output_file>` or `./run.sh 2 <train_data> <test_data> <output_file>`

Scaled the feature vectors in range [0, 1]

(a) Binary Classification

Since d = 8, subset of training data is selected for the images of article class 8 and 9. Implemented the SVM dual problem with a linear kernel and a gaussian kernel.

Objective Function:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \{1 - y^{(i)} \cdot (w^T x^{(i)} + b)\}$$

Observations	Linear Kernel	Gaussian Kernel
Training Accuracy	100%	100%
Validation Accuracy	99.9%	99.8%
Test Accuracy	99.9%	99.8%
Training Time(s)	89 support vectors of 4500	1252 suport vectors of 4500

(b) Multi-Class Classification

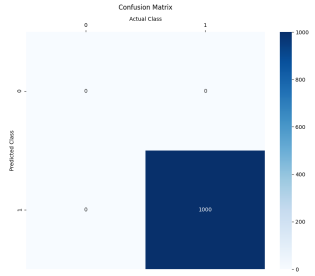
There are total 10 classes so total number of classifiers are 45. The classifiers are trained using Gaussian Kernel. The CVXOPT implementation has been compared with the Scikit Library Implementation below.

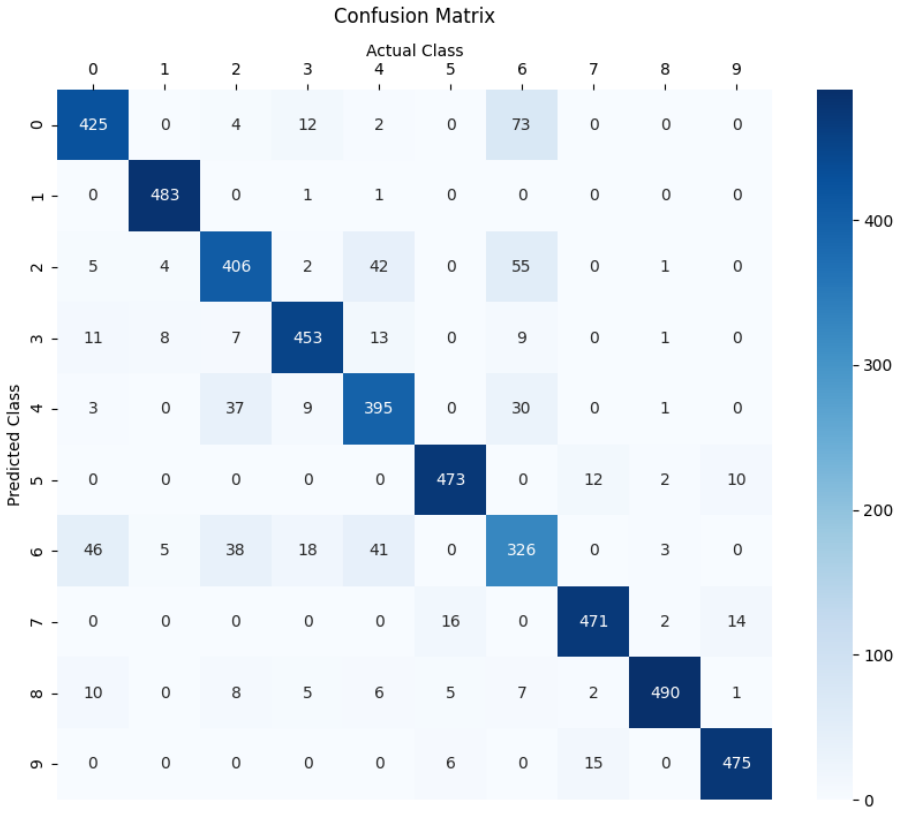
Observations	Multi-Class CVXOPT SVM	Scikit Library SVM
Training Accuracy	96.96%	96.92%
Validation Accuracy	87.94%	88.08%
Test Accuracy	87.94%	88.08%
Training Time(s)	1159.21	299.1
Training Prediction time(s)	42.51	315.6
Validation Prediction Time(s)	12.40	78.22
Test Prediction Times(s)	11.42	78.22

Conclusion: Scikit Implementation is much faster than CVXOPT but accuracies obtained by both are almost same.

(c) Confusion Matrix

1. `True Positives`: 4397
2. `False Positives`: $[500, 500, 500, 500, 500, 500, 500, 500, 500, 500]$
3. `False Negatives`: $[516, 485, 515, 502, 475, 497, 477, 503, 534, 496]$



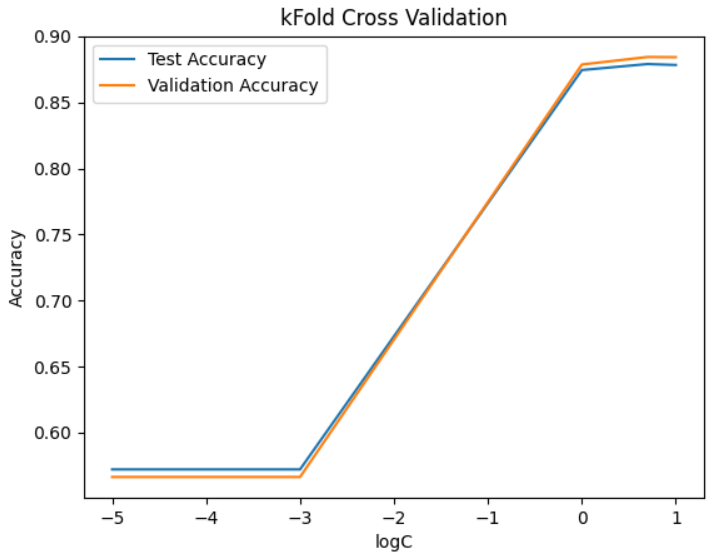


Observation: The classifier performs very well as can be observed from the diagonal entry Class 6 is mis-classifed most, Shirts are mis-classified into T-Shirts most often(73 labels).

(d) **K-fold Cross Validation**: 5-Fold Cross Validation has been performed with $\gamma = 0.05$ and $C \in \{10^{-5}, 10^{-3}, 1, 5, 10\}$ to estimate the best value of C parameter.

Observations(5-Fold)	**C=10**	**C=5**	**C=1**	**C=10−3**	**C=10^{-5}**
Training Accuracy	100%	99.975%	96.965%	57.082%	57.082%
Validation Accuracy	88.426%	88.44%	87.872%	56.644%	56.644%
Test Accuracy	87.836%	87.908%	87.452%	57.22%	57.22%

```
(col333) sharks@sharique:~/Desktop/SEM5/COL774/Assignments/Assignment2/Q2$ python3 kFoldSVM.py
Data Loaded
C = 10
Train Acc =  [1. 1. 1. 1. 1.]
Val Acc =  [0.87888889 0.884      0.89088889 0.88422222 0.88333333]
Test Acc =  [0.877  0.8772 0.8802 0.8788 0.8786]
C = 5
Train Acc =  [0.99988889 0.99977778 0.99972222 0.99972222 0.99966667]
Val Acc =  [0.88044444 0.884      0.89088889 0.88422222 0.88244444]
Test Acc =  [0.8784 0.8776 0.8814 0.8786 0.8794]
C = 1
Train Acc =  [0.97005556 0.96927778 0.97027778 0.96966667 0.969     ]
Val Acc =  [0.878      0.87688889 0.88488889 0.87244444 0.88133333]
Test Acc =  [0.876  0.874  0.8758 0.873  0.8738]
C = 0.001
Train Acc =  [0.57472222 0.56155556 0.57894444 0.57255556 0.56633333]
Val Acc =  [0.55866667 0.55377778 0.57711111 0.57622222 0.56644444]
Test Acc =  [0.5726 0.5612 0.5828 0.5758 0.5686]
C = 1e-05
Train Acc =  [0.57472222 0.56155556 0.57894444 0.57255556 0.56633333]
Val Acc =  [0.55866667 0.55377778 0.57711111 0.57622222 0.56644444]
Test Acc =  [0.5726 0.5612 0.5828 0.5758 0.5686]
logC =  [ 1.       0.69897  0.      -3.      -5.     ]
kFoldTrainAcc =  [1.0, 0.9997555555555555, 0.9696555555555557, 0.5708222222222222, 0.5708222222222222]
kFoldValAcc =  [0.8842666666666666, 0.8844, 0.8787111111111111, 0.5664444444444444, 0.5664444444444444]
kFoldTestAcc =  [0.87836, 0.8790800000000001, 0.8745200000000001, 0.5721999999999999, 0.5721999999999999]
Time =  9543.575798511505
(col333) sharks@sharique:~/Desktop/SEM5/COL774/Assignments/Assignment2/Q2$
```



Conclusion: From the graph it can be seen that where we get high cross-validation accuracy we also get high test accuracy indicating that Cross-Validation is a good metric for hyperparameter tuning. $C = 5$ and $C = 10$ gives the best value of accuracies. For lower values of C the model underfits the data which is why we get very less accuracies. Best value of 5-Fold Cross validation accuracy is 88.44% for $C = 5$, For this value of C, Test Set accuracy is 87.908% which is also the best value of Test Set Accuracy among all C.

3 Large Scale Text Classification

(a) **LIBLINEAR**: Hyperparamter tuning has been done to find the best value of C on validation set.

Observations	**C=10**	**C=5**	**C=1**	**C=10−1**	**C=10^{-2}**
`Training Time(s)`	1592.65	1636.618	147.655	1152.695	1036.925
`Training Accuracy`	78.88%	77.95%	81.696%	77.874%	72.07%
`Validation Accuracy`	61.43%	61.20%	63.74%	65.433%	66.35%
`Test Accuracy`	62.08%	61.596%	64.102%	65.822%	66.617%

Observation: Best value of $C = 10^{-2}$ which gives the highest validation and test accuracies.

```
C =  10
/home/sharks/.local/lib/python3.8/site-packages/sklearn/svm/_base.p
  warnings.warn("Liblinear failed to converge, increase "
Training Time =  1592.6594452857971
Training Accuracy =  0.7888234756452229
Val Accuracy =  0.6143434041280287
Test Accuracy =  0.6208588222976712
C =  5
Training Time =  1636.6187753677368
Training Accuracy =  0.7795003573031094
Val Accuracy =  0.6120438229135508
Test Accuracy =  0.6159604540899505
C =  1
Training Time =  1447.6551764011383
Training Accuracy =  0.8169652501952703
Val Accuracy =  0.6374887825306611
Test Accuracy =  0.6410206554091447
C =  0.1
Training Time =  1152.6959326267242
Training Accuracy =  0.7787462815548502
Val Accuracy =  0.6543336823212683
Test Accuracy =  0.6582210323217518
C =  0.01
Training Time =  1036.9252128601074
Training Accuracy =  0.7207634653415984
Val Accuracy =  0.6635133113969488
Test Accuracy =  0.6661780762500187
C =  0.001
Training Time =  267.57971692085266
Training Accuracy =  0.6836454888405099
Val Accuracy =  0.6616624289560276
Test Accuracy =  0.6654751043240252
C =  0.0001
Training Time =  95.94953680038452
Training Accuracy =  0.652921576122181
Val Accuracy =  0.6454718815435238
Test Accuracy =  0.6482522921371843
C =  1e-05
Training Time =  63.83846855163574
Training Accuracy =  0.6054999750718761
Val Accuracy =  0.6010507029614119
Test Accuracy =  0.6053036988288787
```

(b) **SGD Classifer**: Hyperparamter tuning has been done to find the best value of alpha on validation set.

Observations	**alpha**=10^{-5}	**alpha**=10^{-4}	**alpha**=10^{-3}	**alpha**=10^{-2}	**alpha**=10^{-1}
`Training Time(s)`	183.984	83.74	71.94	103.83	41.133
`Training Accuracy`	72.745%	68.782%	67.072%	63.877%	57.734%
`Validation Accuracy`	66.156%	67.175%	66.155%	63.369%	57.240%
`Test Accuracy`	66.524%	67.403%	66.359%	63.745%	57.805%

Observation: Best value of $alpha = 10^{-4}$ which gives the highest validation and test accuracies.

Conclusion: SGDClassifier is very fast as compared to LinearSVC. It almost take 10 times less. Accuracies obtained by SGDClassifier are also better than LinearSVC. Thus suggesting that SGDClassifier is better than LinearSVC.

Comparison with Naive Bayes: The accuracy obtained by Naives Bayes in Question 1 is 60.157, and using SVM approach to do text classification is much faster and gives better accuracy.

```
sharks@sharique:~/Desktop/SEM5/COL774/Assignments/Assignment2/Q3$ python3 Q3.py
alpha =  1e-05
Training Time =  183.98411393165588
Training Accuracy =  0.7274545892676118
Val Accuracy =  0.6615689500448699
Test Accuracy =  0.6652432731569422
alpha =  0.0001
Training Time =  83.74059057235718
Training Accuracy =  0.6878251042826516
Val Accuracy =  0.6617559078671852
Test Accuracy =  0.6640392467730597
alpha =  0.001
Training Time =  71.94416403770447
Training Accuracy =  0.6707264886244662
Val Accuracy =  0.6615502542626384
Test Accuracy =  0.6635980197131276
alpha =  0.01
Training Time =  103.83057904243469
Training Accuracy =  0.6387727884599405
Val Accuracy =  0.6336935387376608
Test Accuracy =  0.6374534468059648
alpha =  0.1
Training Time =  41.133201360702515
Training Accuracy =  0.5773457364598741
Val Accuracy =  0.5724087645827102
Test Accuracy =  0.5780523190595133
alpha =  1
Training Time =  44.444910287857056
Training Accuracy =  0.49708133215894174
Val Accuracy =  0.4935686509123542
Test Accuracy =  0.49781629997457333
alpha =  5
Training Time =  61.08273410797119
Training Accuracy =  0.44449545477207386
Val Accuracy =  0.4410521986239904
Test Accuracy =  0.4454074993643339
alpha =  10
Training Time =  38.74859356880188
Training Accuracy =  0.43990660262908615
Val Accuracy =  0.4372382590487586
Test Accuracy =  0.44077087602267456
```