COL774 Assignment 2

Navneel Singhal

December 3, 2020

Contents

1 Text Classification

1.1 Implementing a Naive Bayes Classifier

We use the Multinoulli event model for this problem, which is a bag of words model. Since the model doesn't care about the location of words (as we will see below), we will represent the document as a dictionary from words to their frequency in the document. The vocabulary is created while training the model.

Note that parameters in the model are $\{\phi_k\}_{k=1}^r$ and $\theta_{j=l|k}$, whose expressions are given as follows:

$$\phi_k = \frac{\sum_{i=1}^m 1\{y^{(i)} = k\}}{m} = \frac{\text{number of documents of class } k}{m}$$

$$\theta_{j=l|k} = \frac{\sum_{i=1}^m 1\{y^{(i)} = k\} \sum_{i=1}^m 1\{x_j^{(i)} = l\}}{\sum_{i=1}^m 1\{y^{(i)} = k\} n^{(i)}} = \frac{\text{total number of occurences of word } j \text{ in all documents of class } k}{\text{total number of words in all documents of class } k}$$

From here, we see that the representation of our data as a frequency dictionary is sufficient for computation of these parameters.

1. Accuracy on the test set = 60.15%.

2. Accuracy on the train set = 63.11%.

1.2 Random and majority predictions

1. The probability of a guess being correct is $\frac{1}{r}$ where r is the number of classes, so the expected number of correct guesses is $\frac{m}{r}$. Thus the accuracy would be $\frac{100}{r}\%$, which in this case is 20%. Upon a simulation, the accuracy comes out to be 19.95%, which is reasonably close to this answer.

2. The probability of a guess being correct is the fraction of the documents in the test set which are of the class which has the highest frequency in the training set. The accuracy is thus 100 times this fraction, which is 43.99%. Note that if we can assume that the fraction of this class is the same in all test sets (i.e., all test sets are generated by the same ϕ parameters), then the answer for all the test sets will be the same.

As far as the improvement is concerned, the accuracy increase is about 40 and 16 percentage points respectively.

1.3 Confusion matrix

The definition of the confusion matrix I will be using is the following: c_{ij} consists of the number of documents of class j classified into class i by the classifier.
The confusion matrix for the test data is as follows:

$$\begin{bmatrix} 14318 & 2784 & 1349 & 1091 & 3143 \\ 3861 & 3318 & 1695 & 725 & 321 \\ 1174 & 3325 & 5238 & 2549 & 609 \\ 451 & 1067 & 5339 & 18009 & 15191 \\ 365 & 344 & 910 & 6984 & 39558 \end{bmatrix}$$

The class corresponding to 5 star reviews has the diagonal entry. This is because even though there is a significant amount of misclassification in this class (a lot of these are misclassified into the class corresponding to 4 star reviews), the number of documents of this class are high both in the test and the training data.

Some other observations are as follows:

1. There is a significant amount of misclassification of class 2 samples into class 3 and of class 3 samples to class 4. This is probably because of the higher number of samples in class 3 as compared to class 2 and in class 4 compared to class 3, in the training data.

2. There is a significant amount of confusion in class 4 and class 5. This is because the words that are used in a 5 star review might be very similar to those in a 4 star review.

3. Quite a few 2 star, 4 star and 5 star reviews are misclassified into 1 star reviews. This is because of the large number of 1 star reviews (which are just after 4 and 5 star reviews) in the training data, and the reason why 2 star reviews were misclassified more than 3 star reviews is that 2 star reviews have similar sentiments to 1 star reviews, while 3 star reviews might be more of average reviews.

1.4 Stemming and Stopword Removal

The accuracy reduces by a small amount, to 59.64% on the test set. Just for reference, the confusion matrix is as follows:

$$\begin{bmatrix} 14296 & 2919 & 1473 & 1226 & 3251 \\ 3801 & 3219 & 1624 & 639 & 328 \\ 1097 & 2999 & 4966 & 2457 & 587 \\ 491 & 1207 & 5266 & 17473 & 14853 \\ 484 & 494 & 1202 & 7563 & 39803 \end{bmatrix}$$

As we can see, there is no significant difference from the previous confusion matrix. Some of the reasons could be as follows:

1. The text itself is quite small and removal of stop words might not affect the model in a significant way.

2. Stemming might be too strong and might result in some inconsistencies with different forms of the word as well as treating words with different meanings the same.

However, when using the WordNet lemmatizer, the accuracy was nearly the same.

1.5 Feature engineering

The two features used were bigrams and logarithmically scaled frequency. This is motivated from the fact that if there are two words with very high frequencies, but with a ratio comparable to 1, then the difference between them is not as important as that between two words with low frequencies with the same ratio. Hence it makes sense to dampen out the effect of frequencies of words with a high frequency using logarithms.

As mentioned in 1.1, we use a bag of words model with a dictionary containing frequencies. Then the frequencies are transformed under the transformation $f \mapsto \log_2(1 + f)$, where the 1 is to make sure $f = 0, 1$ are handled correctly.

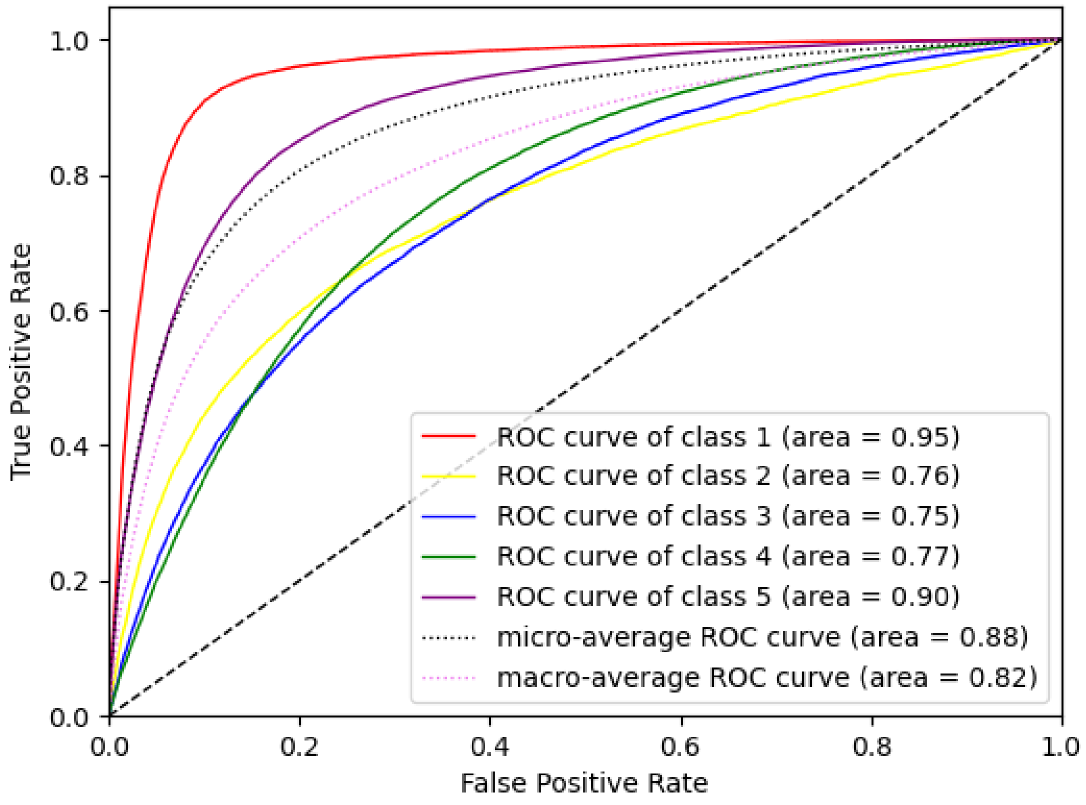
The bigram model is simple; instead of words, we concatenate two consecutive words after stemming and stopword removal and consider them to be a single word, and use the original model on that after transforming frequencies as specified above. This model should perform better than a vanilla model because it takes into account the relative positions of words which exploits more information from the text. However one drawback is that the probability of a bigram occuring is not that high, and hence it might not perform as well on smaller texts.

The accuracy using both these feature transformations was 63.2%. The accuracy using only the bigram features was 63.18%, and the accuracy using logarithmically scaled frequency was 60.22%. Note that all of these features increase the accuracy from 59.64% and thus all these features are important, and these improvements are mainly because of the explanations above.

Note that the best model submitted doesn't use stemming, since stemming reduces the classification accuracy, and the accuracy of this model is 64.27%.

1.6 Receiver Operating Characteristic Curves

As mentioned, the ROC curves have been plotted and they are as follows:



Explanation

1. The shape of the ROC curve shows that as we increase the false positive rate, the true positive rate also increases, and this implies the existence of a trade-off between FPR and TPR.
2. Another noticeable observation is that these graphs are concave, and thus the rate of increase of TPR decreases as we increase FPR.
3. A possible choice for the thresholds could be where the TPR equals the area under the curve (AUC), or a point where the TPR and the FPR sum to 1 (so true negative rate = true positive rate).
4. The steepest curve is for class 1, and this can be explained using the confusion matrix, where the ratio of misclassifications from and to class 1 was quite less. Then comes class 5, and the classes 2, 3 and 4 are comparable, again due to a similar explanation.
5. Microaveraging computes the TPR and FPR treating all samples with equal importance, while macroaveraging does this within each class, and then averages over classes, treating all classes with equal importance. The fact that these are different suggests that there is class imbalance, and this is clearly the case if we look at the confusion matrix for the test data (by adding along the columns).

2 Fashion MNIST Article Classification

2.1 Binary Classification

My entry number ends with 0, so $d = 0$, and thus we will consider binary classification with classes $0, 1$.

2.1.1 Linear Kernel

Mathematical Formulation Note that the SVM formulation has to minimize $-W(\alpha)$ which is

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)\intercal}x^{(j)} - \sum_{i=1}^{m}\alpha_i$$

subject to the constraints

$$0 \le \alpha_i \le C \ \forall i \in \{1, 2, \dots, m\} \quad \text{and} \quad \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

Note that in the notations of this link, we need to rephrase this problem into the following form:

$$\min_{\alpha} \tfrac{1}{2}\alpha^\intercal P\alpha + q^\intercal\alpha$$
$$\text{subject to } G\alpha \preceq H \text{ and } A\alpha = b.$$

Define Z to be the matrix such that $Z_{ij} = x_j^{(i)}y^{(i)}$.
Then we have

$$\begin{aligned}
\alpha^\intercal P\alpha &= \alpha^\intercal\left(P\alpha\right)\\
&= \begin{bmatrix}\alpha_1 & \alpha_2 & \dots & \alpha_m\end{bmatrix}\begin{bmatrix}\sum_{j=1}^{m}P_{1j}\alpha_j\\ \sum_{j=1}^{m}P_{2j}\alpha_j\\ \vdots\\ \sum_{j=1}^{m}P_{mj}\alpha_j\end{bmatrix}\\
&= \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j P_{ij}
\end{aligned}$$

Comparing $\frac{1}{2}\alpha^\intercal P\alpha$ to the first term in $W(\alpha)$, we have $P_{ij} = \left(y^{(i)}x^{(i)}\right)^\intercal\left(y^{(j)}x^{(j)}\right)$, which is precisely $\left(ZZ^\intercal\right)_{ij}$, which means $P = ZZ^\intercal$.

Note that $q^\intercal\alpha = \sum_{i=1}^m q_i\alpha_i$, so we have $q_i = -1$ for all i.

To write the constraints, we note that the constraints can be written as $-\alpha_i \le 0$ and $\alpha_i \le C$. If G has dimension $g \times m$, then we have $\sum_{i=1}^m g_{ji}\alpha_i \le H_j$ for all $j \in \{1, 2, \dots, g\}$.

To represent the constraints, we can have $g = 2m$, with the following happening: if $j \le m$, then $g_{ji} = -1$ if $j = i$ and 0 otherwise, and if $j > m$, $g_{ji} = 1$ if $j = i$ and 0 otherwise. Note that then we must have $H_i = 0$ for $i \le m$ and $H_i = C$ for $i > m$.

To write out the equality constraints, we have $A = y^\intercal$ and $b = 0$.
All in all, we have the following definitions:

$$Z_{ij} = x_j^{(i)}y^{(i)} \qquad P = ZZ^\intercal \qquad q = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

$$G = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \qquad h = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ C \\ C \\ \vdots \\ C \end{bmatrix} \qquad A = y^\intercal \qquad b = 0$$

Note that the value of c as mentioned in the assignment spec (note that this is not C) doesn't need to be taken into consideration since the argmin doesn't depend on c.
w and b can be represented simply as a vector and a scalar respectively, and w has the same shape as the training example.

Results

1. There are 198 support vectors.
2. The test accuracy is 98%.
3. The validation accuracy is 97.8%.

2.1.2 Gaussian Kernel

Mathematical Formulation Note that the form of $-W(\alpha)$ changes to the following:

$$\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y^{(i)}y^{(j)}\phi\left(x^{(i)}\right)^\intercal \phi\left(x^{(j)}\right) - \sum_{i=1}^m \alpha_i$$

The constraints are the same as in the previous part, and since the coefficients of α_i are the same, q is also the same as the previous part.

So we can focus all our attention to getting the new form of P. By a similar analysis as in the previous part, we have $P_{ij} = y^{(i)}y^{(j)}\phi\left(x^{(i)}\right)^\intercal \phi\left(x^{(j)}\right) = y^{(i)}y^{(j)}\exp\left(-\gamma\|x^{(i)} - x^{(j)}\|^2\right)$. This matrix can be found by the Hadamard multiplication of the matrix $yy^\intercal$ and the matrix M such that $M_{ij} = \exp\left(-\gamma\|x^{(i)} - x^{(j)}\|^2\right)$. For this, we compute the matrix J such that $J_{ij} = \|x^{(i)} - x^{(j)}\|^2$; since computing M from this would be straightforward.

We have $J_{ij} = \|x^{(i)}\|^2 + \|x^{(j)}\|^2 - 2x^{(i)\intercal}x^{(j)}$. Note that we can split J into three parts J_1, J_2, J_3, each corresponding to each term in the summation. The computation of the first two parts is straightforward, and for computing the last part, note that it is equal to $-2XX^\intercal$ where X is the training matrix.

Now note that for testing, we can't explicitly store w, which is an infinite-dimensional vector. Hence, while predicting, we always use the whole set of support vectors, which have their corresponding $\alpha \neq 0$. I have implemented a vectorized form of testing (which will be used in the next part) which makes use of a similar sort of a matrix as P in the previous part, where I have $J_{ij} = \|x^{(i)} - x'^{(j)}\|^2$, where x' is the test set and x is the set of all support vectors of the model.

Results

1. There are 858 support vectors.
2. The training accuracy is 98.9%.
3. The validation accuracy is 98.6%.

These are strictly better than those found using the linear kernel. One possible explanation about why this is better is that according to this paper, the linear kernel is a degenerate case of the RBF kernel, and thus the optimal RBF kernel (with the best C) will be always better than the optimal linear kernel (with the best C for any linear kernel).

2.2 Multi-Class Classification

2.2.1 Training using our SVM implementation

Using the given values of C, γ, the test accuracy is 88.08% and the validation accuracy is 87.92%.

2.2.2 Training using `sklearn`'s SVM implementation

Using the given values of C, γ, the test accuracy is 88.08% and the validation accuracy is 87.92%. These are precisely the same as those found in the previous part.

To compare the computational cost, we note the following running times:

	Training time	Test prediction time	Validation prediction time
Our implementation	1061.03s	8.15s	3.94s
`sklearn`'s implementation	224.72s	53.89s	26.76s

As can be noted from a simple profiling of the program, the most time is taken during the solving of the dual optimization problem using `CVXOPT`, and use of a different quadratic optimization solver will drastically improve the training time.

2.2.3 Confusion Matrix

Confusion matrix for test data using own implementation

$$\begin{bmatrix}
433 & 1 & 5 & 12 & 3 & 0 & 80 & 0 & 1 & 0 \\
0 & 482 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 4 & 411 & 3 & 41 & 0 & 55 & 0 & 1 & 0 \\
11 & 9 & 7 & 457 & 13 & 0 & 9 & 0 & 1 & 0 \\
3 & 0 & 37 & 9 & 399 & 0 & 34 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 473 & 0 & 14 & 2 & 11 \\
38 & 4 & 32 & 14 & 38 & 0 & 315 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 16 & 0 & 471 & 2 & 14 \\
10 & 0 & 8 & 5 & 5 & 5 & 7 & 1 & 489 & 1 \\
0 & 0 & 0 & 0 & 0 & 6 & 0 & 14 & 0 & 474
\end{bmatrix}$$

Confusion matrix for validation data using own implementation

$$\begin{bmatrix}
212 & 0 & 5 & 6 & 1 & 0 & 34 & 0 & 0 & 0 \\
0 & 237 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 3 & 206 & 0 & 24 & 0 & 28 & 0 & 1 & 0 \\
8 & 7 & 3 & 228 & 8 & 1 & 3 & 0 & 1 & 0 \\
0 & 0 & 18 & 6 & 200 & 0 & 19 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 241 & 0 & 8 & 0 & 6 \\
26 & 2 & 13 & 9 & 15 & 0 & 165 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 230 & 2 & 8 \\
3 & 1 & 5 & 1 & 1 & 1 & 1 & 1 & 244 & 1 \\
0 & 0 & 0 & 0 & 0 & 5 & 0 & 11 & 0 & 235
\end{bmatrix}$$

Confusion matrix for test data using `sklearn` implementation

$$\begin{bmatrix}
433 & 1 & 5 & 12 & 3 & 0 & 80 & 0 & 1 & 0 \\
0 & 482 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 4 & 411 & 3 & 41 & 0 & 55 & 0 & 1 & 0 \\
11 & 9 & 7 & 457 & 13 & 0 & 9 & 0 & 1 & 0 \\
3 & 0 & 37 & 9 & 399 & 0 & 34 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 473 & 0 & 14 & 2 & 11 \\
38 & 4 & 32 & 14 & 38 & 0 & 315 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 16 & 0 & 471 & 2 & 14 \\
10 & 0 & 8 & 5 & 5 & 5 & 7 & 1 & 489 & 1 \\
0 & 0 & 0 & 0 & 0 & 6 & 0 & 14 & 0 & 474
\end{bmatrix}$$

Confusion matrix for validation data using `sklearn` implementation

$$\begin{bmatrix}
212 & 0 & 5 & 6 & 1 & 0 & 34 & 0 & 0 & 0 \\
0 & 237 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 3 & 206 & 0 & 24 & 0 & 28 & 0 & 1 & 0 \\
8 & 7 & 3 & 228 & 8 & 1 & 3 & 0 & 1 & 0 \\
0 & 0 & 18 & 6 & 200 & 0 & 19 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 241 & 0 & 8 & 0 & 6 \\
26 & 2 & 13 & 9 & 15 & 0 & 165 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 230 & 2 & 8 \\
3 & 1 & 5 & 1 & 1 & 1 & 1 & 1 & 244 & 1 \\
0 & 0 & 0 & 0 & 0 & 5 & 0 & 11 & 0 & 235
\end{bmatrix}$$

Results

1. As can be seen from the results, the diagonal element is almost always more than an order of magnitude higher than any other element in the same column (and many of the off-diagonal elements are 0). This implies that the model has a high accuracy.

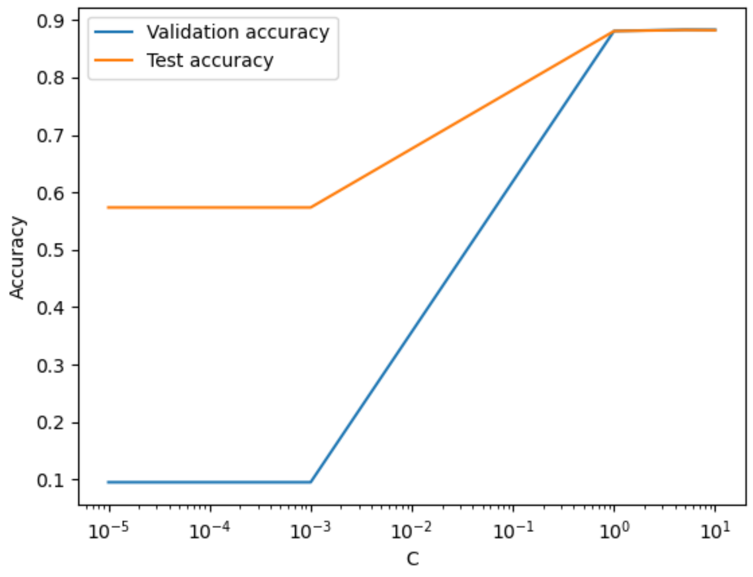
2. Classes 6 and 4 are the two classes which have the highest rate of getting misclassified into other classes. From a quick lookup on the link provided, these are the classes corresponding to shirts and coats. Shirts are most frequently misclassified as t-shirts/tops, pullovers and coats, and coats are most frequently misclassified into pullovers and shirts.

3. This makes intuitive sense because they have the most similar structures visually, which translate into similar feature vectors, since the darkness of all pixels determines the image.

2.2.4 k-fold cross validation

For the purpose of this assignment, two strategies have been used. Firstly, we perform a 5-fold cross validation for each value of C, and the results are tabulated as follows:

C	10^{-5}	10^{-3}	1	5	10
Validation accuracy	9.51%	9.51%	88.09%	88.33%	88.32%
Test accuracy	57.36%	57.36%	88.08%	88.28%	88.24%



Then for the submission, to make the selection more efficient, we use a validation set instead of k-fold cross validation and reduce the search space of C down to $\{1, 5, 10\}$. The results remain pretty much the same.

Observations

1. Note that the results for $C = 10^{-5}, 10^{-3}$ are pretty bad, and are almost as bad as a random classification, and for larger C values, the accuracy saturates at about $C = 5$. In the case of higher C-values, the classification is closer to a hard-margin SVM, and for smaller C-values, it tries to maximize the margin even at the cost of misclassifications (since C is small, there is more room for errors). Hence we can conclude that the data is almost perfectly separable when considering two classes. It can also be seen from the fact that in the previous parts, the binary classification was almost perfect.
2. Note that $C = 5$ gives both the best validation accuracy as well as the best test accuracy. However, there is an accuracy saturation when we consider $C > 1$, so we do not hardcode parameters, but do hyperparameter tuning in the final set as well.

3 Large scale text classification

The preprocessing of the text was the same for all models - `CountVectorizer` and `TfidfTransformer`. The latter of these has a similar logarithmic effect on the frequency as described in the first question.

3.1 Comparing `sklearn`'s Naive Bayes and SVM with `LIBLINEAR`

Note that `LinearSVC` has been used here, and the parameter tuned in hyperparameter tuning is C, which is the regularization parameter.

Hyperparameter Tuning for `LinearSVC`

C	1	3	5
Validation accuracy	66.51%	66.52%	66.28%

Now to compare this to Naive Bayes, the value of C with the maximum validation accuracy was used in training the whole model, and the results are as follows:

Comparison between models

Model	Accuracy	Training Time
Naive Bayes	60.04%	85.71s
Linear SVM	66.38%	470.61s

Note that the training time for the SVM also includes time taken in training the validation models.

3.2 Comparing `sklearn`'s SVM with `LIBLINEAR` and SGD Classifier with Hinge Loss

For the SGD classifier, we tune the hyperparameter α, which is the SGD regularization parameter.

Hyperparameter tuning for SGD classifier

α	10^{-6}	2×10^{-6}	5×10^{-6}	10^{-5}
Validation accuracy	66.41%	66.67%	66.11%	65.69%

Now to compare this to the SVM with `LIBLINEAR`, the value of α with the maximum validation accuracy was used in training the whole model, and the results are as follows:

Comparison between models

Model	Accuracy	Training Time
Linear SVM	66.38%	470.61s
SGD Classifier	66.97%	163.37s

Note that both training times include time taken in training the validation models.