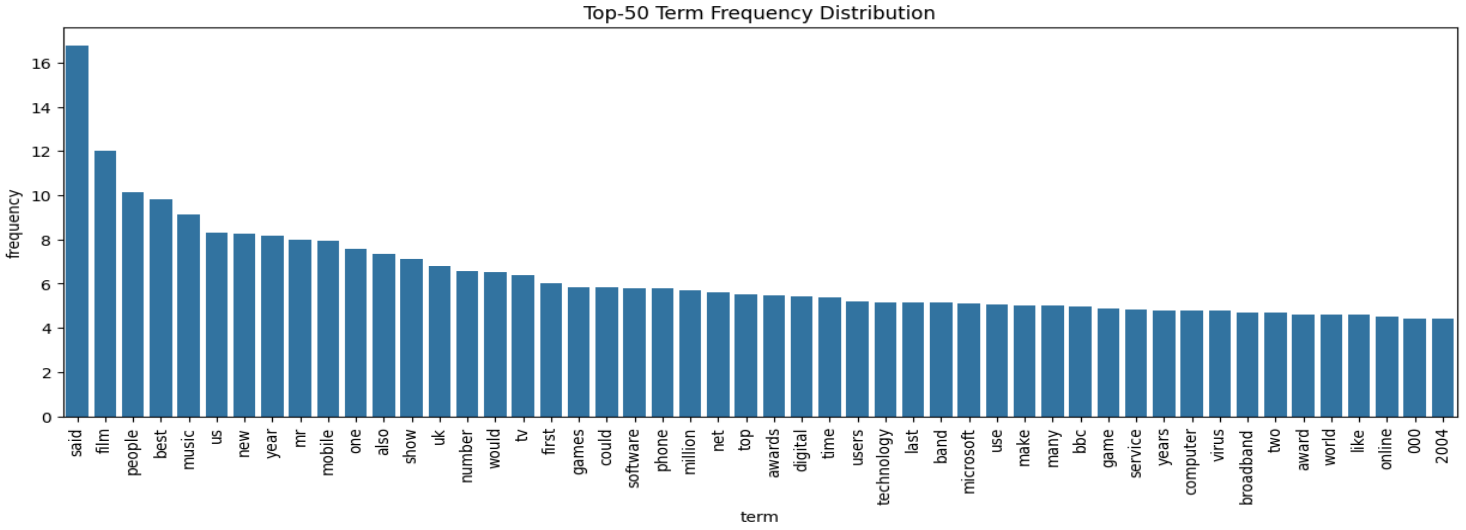
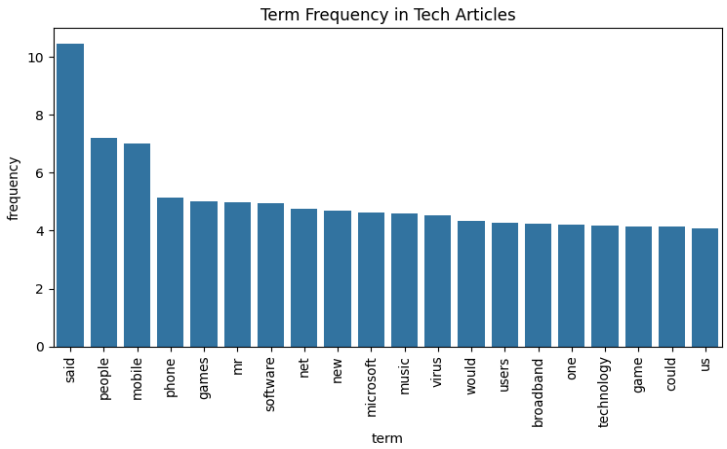
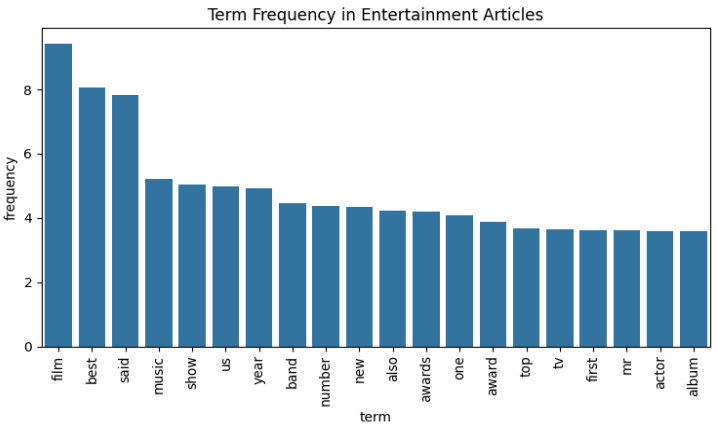
Task 1a&b (please refer to code for more apparent plots)

To extract the features in each article, we used TFIDF (Term Frequency-Inverse Document Frequency). TFIDF measures the importance of words in a document relative to a collection of documents. There were 428 total articles and 13518 extracted features. Five articles and their feature vectors are shown below.

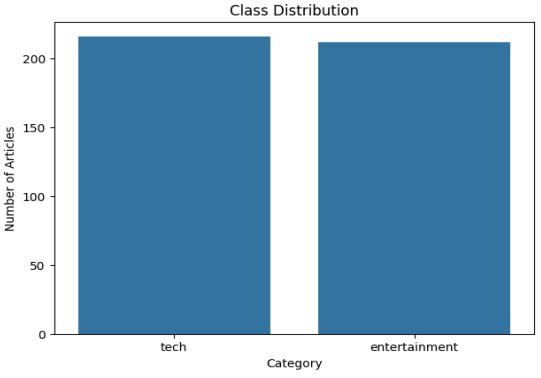
```
Number of articles: 428
Number of extracted features: 13518
   ArticleId   00        000  000th  001st  0051  007  0100  0130  028  ...  \
0       1976  0.0   0.020115    0.0    0.0   0.0  0.0   0.0   0.0  0.0  ...
1       1797  0.0   0.000000    0.0    0.0   0.0  0.0   0.0   0.0  0.0  ...
2       1866  0.0   0.000000    0.0    0.0   0.0  0.0   0.0   0.0  0.0  ...
3       1153  0.0   0.000000    0.0    0.0   0.0  0.0   0.0   0.0  0.0  ...
4        342  0.0   0.000000    0.0    0.0   0.0  0.0   0.0   0.0  0.0  ...

   zola  zombie  zombies  zone  zonealarm  zones  zoom  zooms   zooropa  zorro
0   0.0     0.0      0.0   0.0        0.0    0.0   0.0    0.0  0.000000    0.0
1   0.0     0.0      0.0   0.0        0.0    0.0   0.0    0.0  0.000000    0.0
2   0.0     0.0      0.0   0.0        0.0    0.0   0.0    0.0  0.000000    0.0
3   0.0     0.0      0.0   0.0        0.0    0.0   0.0    0.0  0.000000    0.0
4   0.0     0.0      0.0   0.0        0.0    0.0   0.0    0.0  0.054551    0.0

[5 rows x 13519 columns]
```







- There was roughly an equal number of articles in each class as shown by the bar chart representing the class distribution
- We can see that the distribution of words in all three plots is skewed to the right.
- 'Said' and 'Film' are two words which dominated the distributions, causing the plots to be skewed.
- There were some common words shared across both classes such as new and music, but these are rare since TFIDF prioritises terms that are unique or distinctive to each class.



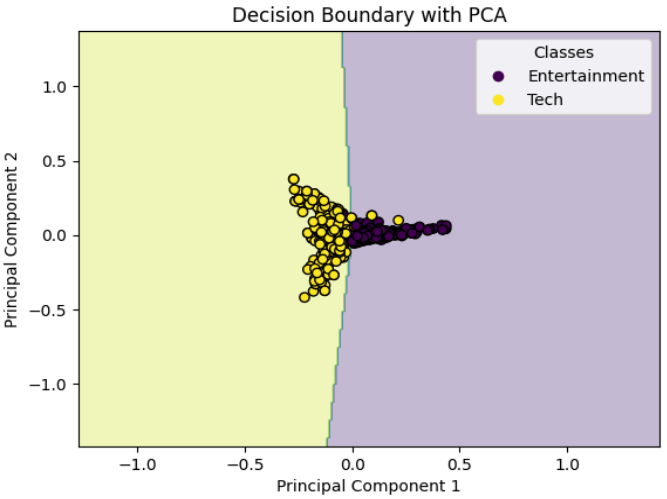
Task 2a

```
Top 20 most identifiable words for each class:
Class entertainment:
['film', 'best', 'said', 'band', 'show', 'music', 'year', 'awards', 'album', 'us', 'star', 'actor', 'top',
'award', 'number', 'tv', 'new', 'one', 'singer', 'oscar']
Class tech:
['said', 'mobile', 'people', 'games', 'software', 'microsoft', 'virus', 'phone', 'net', 'technology', 'mr',
'users', 'game', 'would', 'music', 'use', 'new', 'digital', 'search', 'could']
```

```
Top 20 words that maximize quantity
Class entertainment:
['film', 'band', 'album', 'best', 'actor', 'singer', 'star', 'oscar', 'award', 'musical', 'chart', 'rock',
'theatre', 'comedy', 'stars', 'elvis', 'song', 'festival', 'awards', 'nominations']
Class tech:
['mobile', 'software', 'microsoft', 'games', 'virus', 'technology', 'users', 'net', 'phone', 'broadband',
'phones', 'spyware', 'computer', 'search', 'pc', 'firms', 'use', 'mail', 'data', 'game']
```

The second list of words describes each class better. This is because, in the first list, there were words like 'could,' 'Mr', 'said,' etc., which have no objective relation to the classes, whereas, in the second list of words, the words are what is most likely to occur in the classes and therefore are much more related to the class and so describe them better. The second one looks at the ratio of words occurring in documents with classy and documents that are not class Y, in contrast to just looking at the most occurring words in each document class. This is important as many documents may have similar occurrences of certain words. Still, certain other words may only occur in one document class, though only sometimes be in the top 20 most occurring words.

Task 2b

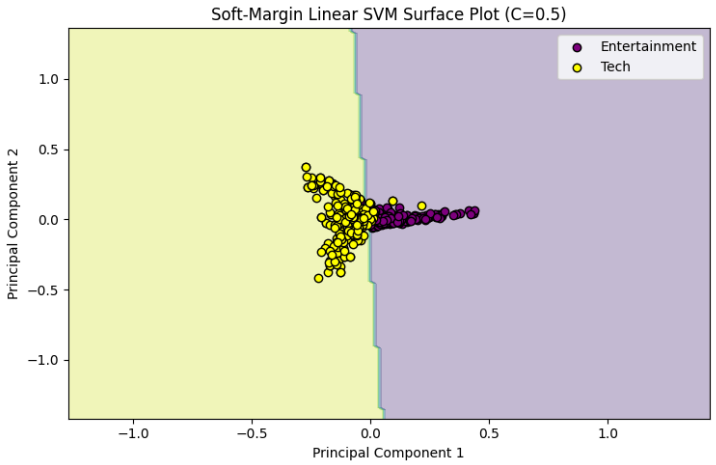


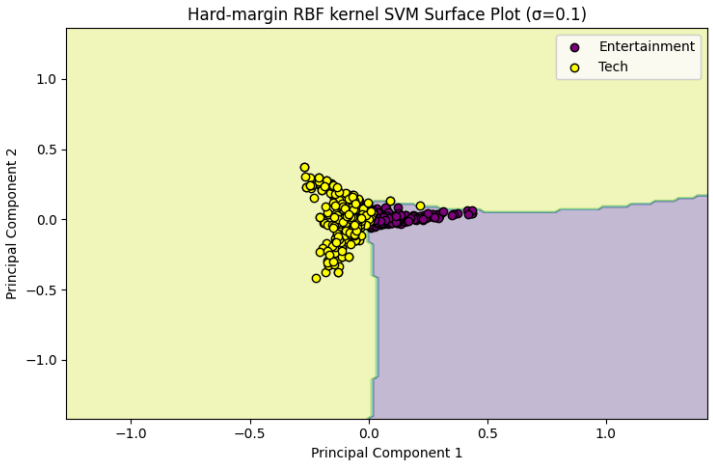
Using an Euclidean distance for the decision boundary shows a high bias toward tech, and similar effects are shown for the Minkowski distance. Using a Manhattan distance for the decision boundary shows a slight bias toward tech. Specifically, we can see a slight protruding circle where the purple data points meet the yellow data points. Using cosine distance for the decision boundary shows an excellent, smooth separation for the two classes and no clear bias for either class. Cosine distance shows a better decision boundary than Euclidean, Minkowski, and Manhattan because cosine distance uses the angle between two vectors, whereas the other three use magnitude. Because there are so many dimensions, the other three distance metrics may suffer the curse of dimensionality, whereas, in high dimensions, the distance between two points becomes similar, making it harder to distinguish between data points. Using a higher k shows a smoother decision boundary and less bias to one class.

Task 2c

The penalty parameter C in a soft-margin SVM controls how misclassifications are penalised. Using a low C value (C=0.5) allows for a wider margin and greater tolerance for misclassifications. In our plot, some tech class points are misclassified, but most points lie within their respective regions. Many points of both classes are near the support vectors, indicating a balanced margin. The sigma value controls the kernel width in our hard-margin SVM. We used gamma for our models, which is inversely related to sigma, as shown in the equation $\gamma = 1/2\sigma^2$. Therefore, our sigma value ($\sqrt{5}$) used is relatively high, suggesting that our SVM is less prone to overfitting and more isolated compared to a smaller sigma, which can be seen by the boundary curving

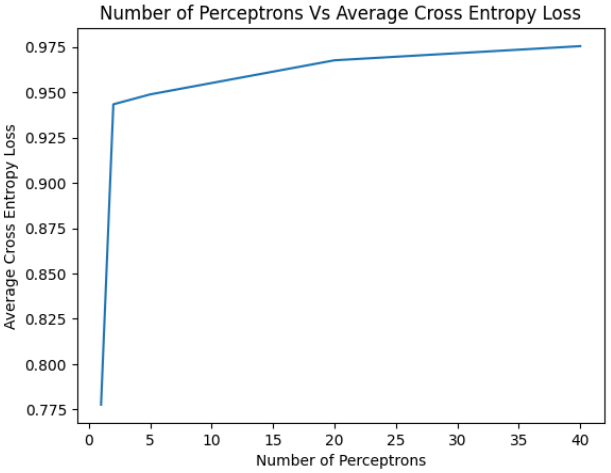
around the class, the entertainment class. From the two plots, it seems that the hard-margin SVM performs better in classifying the examples as there are fewer misclassifications and can fully separate the points.





Task 2d

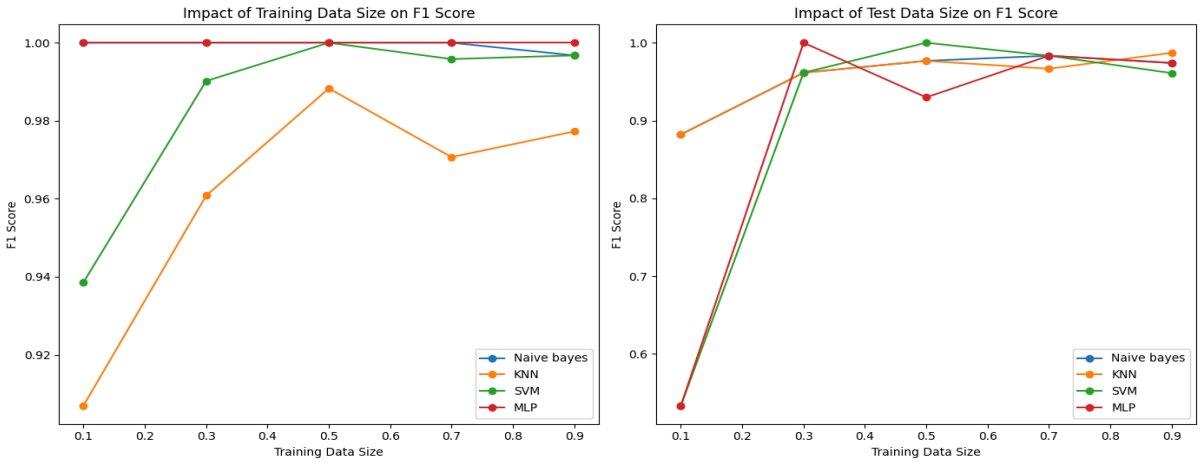
Here is a plot displaying the relationship between the number of perceptrons and cross entropy loss..



- As the number of perceptrons increases, the average cross-entropy loss would increase.
  - When there is only one perceptron, the cross entropy is very low. This is because it gave incorrect predictions. Incorrect predictions would reduce cross entropy significantly.
  - When there are two perceptrons, the cross entropy increases drastically compared to the model with only one perceptron. This is because the model started being able to give more correct predictions, although the probabilities for the correct labels were lower than other models with more perceptrons.
- After increasing to a certain number of perceptrons, the cross entropy would increase slowly.
  - When comparing the model with 20 perceptrons to the model with 40 perceptrons, the cross entropy did not increase by much. This is because when the data could already be described well enough with a certain number of perceptrons, adding more perceptrons would not help.

Task 3a

To create our plots, we divided the dataset into partitions for plotting and conducted train-test splits to evaluate classifier accuracies. MLP and Naive Bayes consistently achieved 100% training F1 scores across partition sizes, while SVM and KNN showed lower scores, improving with larger partitions up to 50% and slightly decreasing thereafter. Test accuracies increased across all classifiers, with NB, SVM and MLP starting below 60% and reaching around 98%-99% on the final partition. KNN ended with the highest F1 score at 90%, followed by MLP and Naive Bayes tied, then SVM, suggesting that KNN performed the best, followed by MLP and NB tied, followed by SVM. However, note that these scores are based on optimised models from the hyperparameter tuning.



Task 3b

NB

For NB, the hyperparameter being tuned is the alpha value added to the Laplace smoothing. Alpha controls the amount of smoothing to ensure no probability estimate becomes zero, reducing the risk of overfitting in the dataset. In our 5-fold cross-validation, we tested 7 different alpha values in the list [0.01, 0.1, 1.0, 5.0, 10.0, 50.0, 100.0]. After running our 5-fold cross-validation procedure, we listed the average F1 score of each classifier and found that the NB classifier with alpha = 0.1 scored the highest with an F1 score of 0.9837. Classifiers with alpha=0.01 and 1.0 scored closely, both receiving a score of 0.9813 and alpha values greater than 1 continued to decrease down to 0. Therefore, the optimal alpha value found is 0.1, with an average F1 score of 0.9837.

```
F1 Scores for Naive mayes with alpha = 0.01: 0.9813
F1 Scores for Naive mayes with alpha = 0.1: 0.9837
F1 Scores for Naive mayes with alpha = 1.0: 0.9813
F1 Scores for Naive mayes with alpha = 5.0: 0.9743
F1 Scores for Naive mayes with alpha = 10.0: 0.9696
F1 Scores for Naive mayes with alpha = 50.0: 0.9079
F1 Scores for Naive mayes with alpha = 100.0: 0.8036
```

KNN

For Knn, two hyperparameters are being tuned. These are the choice of distance metric and the number of k-neighbours. The distance metric can influence the relative importance of different features, while the number of neighbours affects the bias-variance tradeoff. From the hyperparameter tuning, it seems that the choice of distance metric doesn't matter except for Manhattan (where we got the lowest accuracy of 0.622). We got the same accuracies for Euclidean, Minkowski and cosine distance metrics with k = 16, but in terms of decision boundaries, cosine similarity had the best decision boundary. Therefore, the optimal hyperparameter settings is k=16, metric=cosine

```
Accuracy and k neighbours for euclidean, manhattan, minkowski and cosine
[0.9796248934356353, 0.9328218243819266, 0.9796248934356353, 0.9796248934356353]
[16, 11, 16, 16]
```

SVM

For our 5-fold cross-validation on the SVM model, we decided to use 3 hyperparameters: margin type (linear/RBF), miss classification penalty (C) and kernel width (sigma/gamma). For the linear SVM, we are comparing C-values in [10, 1, 0.5, 0.1] to determine which yields the best average F-1 score. After running our procedure, we can see that C = 0.5 yielded the highest mean F1 score of 0.9906,

contrasting with scores of 0.9883 for C = 10 and 1 and 0.9223 for C = 0.1. For the RBF kernel SVM, we are comparing gamma $\gamma = 1/2\sigma^2$ in [10, 1, 0.5, 0.1]. The highest mean F1 score occurred at gamma = 0.5, resulting in an F1 score of 0.9883. Although the score increased from 10 (0.5217) to 1 (0.9836), we saw little improvement past gamma = 0.5. Therefore, the optimal hyperparameter setting is using a linear SVM with C = 0.5.

```
F1 Scores for Linear SVM with c = 10: 0.9883
F1 Scores for Linear SVM with c = 1: 0.9883
F1 Scores for Linear SVM with c = 0.5: 0.9906
F1 Scores for Linear SVM with c = 0.1: 0.9223
F1 Scores for rbf SVM with gamma = 10: 0.5217
F1 Scores for rbf SVM with gamma = 1: 0.9836
F1 Scores for rbf SVM with gamma = 0.5: 0.9883
F1 Scores for rbf SVM with gamma = 0.1: 0.9883
```

MLP

A multi-layer perceptron classifier has the following hyperparameters: number of hidden layers, number of perceptrons in each hidden layer, number of epochs and learning rate.

In general, the more hidden layers and perceptrons there are, the more flexible the model is when fitting into the data. But, too many hidden layers and perceptrons would cause overfitting. Moreover, the number of epochs and learning rate would also affect whether the optimal weightings could be found. If the learning rate is too large, then the optimal weightings would not be accurate. If the learning rate is too small, it would take a lot of time for training. Also, if the learning rate is small but the number of epochs is limited, the optimal weightings might not be obtained before the maximum number of epochs is reached. Below are the experimental results of how different hyperparameters affect cross-validation scores. (For detailed experimental procedures and explanations, please refer to the coding section.)

Comparing different numbers of hidden layers	Cross validation score for MLP with 1 hidden layers: 0.9796029023237096 Cross validation score for MLP with 2 hidden layers: 0.9412931492793971 Cross validation score for MLP with 3 hidden layers: 0.8300802272764912
Comparing different numbers of perceptrons	Cross validation score for MLP with 1 hidden layers, 1 perceptrons: 0.8714032298884709 Cross validation score for MLP with 1 hidden layers, 20 perceptrons: 0.9766610896479578 Cross validation score for MLP with 1 hidden layers, 40 perceptrons: 0.9795743119119841 Cross validation score for MLP with 1 hidden layers, 60 perceptrons: 0.9796029023237096 Cross validation score for MLP with 1 hidden layers, 80 perceptrons: 0.9766610896479578 Cross validation score for MLP with 1 hidden layers, 100 perceptrons: 0.9796029023237096
Comparing different learning rates	Cross validation score for MLP with learning rate of 10: 0.9679088087881837 Cross validation score for MLP with learning rate of 1: 0.9321281529742453 Cross validation score for MLP with learning rate of 0.1: 0.973807452786916 Cross validation score for MLP with learning rate of 0.01: 0.9766610896479578

From the experimental results, we decided that an MLP with 1 hidden layer, 40 perceptrons and a learning rate of 0.01 would be our best MLP classifier.

Task 3c

For the last part of this report, we will look at how well our optimised models perform on the test data. To do this, we will be training on all the training datasets and then generating the predictions using the data from the test set to assess the performance of our optimised models using F1 scores. From this procedure, we can see that the SVM classifier scored the highest with a 1.0 F1 score. This is followed by the MLP classifier with a score of 0.9783 and then NB at 0.9677. Finally, KNN scored the lowest at 0.9677. These results are the inverse of what we got when comparing the F1 scores across different partitions where KNN scored the highest, and SVM scored the lowest. Therefore, some of the hyperparameters may be causing the models to be overfitted to the training data. However, these scores are still consistent with the overall performance trends observed during the training and testing phases.

```
NB F1 score: 0.9677
KNN F1 score: 0.9670
SVM F1 score: 1.0000
MLP F1 score: 0.9783
```